



UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II
CORSO DI LAUREA IN INGEGNERIA INFORMATICA
CORSO DI INGEGNERIA DEL SOFTWARE
PROF. A.R. FASOLINO - A.A. 2024 - 25



PoetUp

“Sistema software: Piattaforma di Condivisione di Poesie Brevi”

Basile Giovanni - N46008729

Canonico Marco - N46007287

Capasso Matteo - N46007050

Carraturo Luca - N46007197

INDICE

1. Specifiche Informali	4
2. Analisi e specifica dei requisiti	6
2.1 Analisi nomi-verbi	6
2.2 Revisione dei requisiti	7
2.3 Glossario dei termini	9
2.4 Classificazione dei requisiti	10
2.4.1 Requisiti funzionali	10
2.4.2 Requisiti sui dati	11
2.4.3 Vincoli / Altri requisiti	12
2.5 Modellazione dei casi d'uso	14
2.5.1 Attori e casi d'uso	14
2.5.2 Diagramma dei casi d'uso	17
2.5.3 Scenari	17
2.6 Diagramma delle classi	26
2.7 Diagramma di sequenza	28
2.7.1 Registrazione	28
2.7.2 Autenticazione	28
2.7.4 PubblicazionePoesia	29
2.7.5 GestioneRaccolte	30
2.7.6 VisualizzaFeed	31
2.7.10 VisualizzaStatistiche	32
2.7.11 RicercaPoesie	33
2.8 Diagramma delle classi raffinato	33
3. Piano di Test Funzionale	34
3.1 Registrazione	34
3.2 Autenticazione	37
3.3 GestioneProfiloPersonale	39
3.4 PubblicazionePoesia	42
3.5 GestioneRaccolte	45
3.5.1 CreazioneRaccolta	45
3.5.2 SpostaPoesiaInRaccolta	46
3.5.3 ModificaRaccolta	47
3.6 CommentoPoesia	48
3.7 GeneraReport	49
3.8 RicercaPoesia	51
3.8.1 RicercaPoesiaPerTag	51
3.8.2 RicercaPoesiaPerTesto	52
3.8.3 RicercaPoesiaPerTitolo	53
4. Progettazione	54
4.1 Class Diagram BCED	54
4.1.1 Traduzione classi ed associazioni	55
4.1.2 Pattern BCED	55
4.1.2.1 Package Boundary	55
4.1.2.2 Package Controller	56
4.1.2.3 Package Entity	56
4.1.2.4 Package Database	57

4.2 Sequence Diagram BCED	58
4.2.3 GestioneProfiloPersonale	60
4.2.3.1 ModificaProfilo	60
4.2.3.2 VisualizzaProfilo	60
4.2.5 GestioneRaccolte	62
4.2.5.1 VisualizzaRaccolta	62
4.2.5.2 CreaRaccolta	62
4.2.7 VisualizzazioneFeed	63
4.2.8 CommentoPoesia	64
4.2.10 GeneraReport	65
4.3 Class Diagram Raffinato	66
5. Implementazione	67
5.1 Package Database	67
5.2 Package Entity	67
5.3 Package Controller	68
5.4 Package Boundary	68
5.4 Package DTO	69
5.5 Package Session	69
5.6 Package Email	69
5.7 Package Facade	69
6. Testing	71
6.1 Test Strutturale	71
6.1.1 Complessità ciclomatica	71
6.1.1.1 like() - EntityPoesia	71
6.1.1.1.1 ScriviSuDB() - ApprezzamentoDAO	72
6.1.1.2 generaReport() - EntityAmministratore	73
6.1.1.2.1 caricaListaDaDB() - EntityPoetUp	74
6.1.1.2.2 convertiPoesieDAO - EntityAmministratore	75
6.1.1.2.3 calcoloTageAutore() - EntityAmministratore	75
6.1.2 Test funzionale	77
6.2.1 Registrazione	78
6.2.2 Autenticazione	79
6.2.3 GestioneProfiloPersonale	81
6.2.4 Pubblicazione Poesia	82
6.2.5 CreazioneRaccolta	85
6.2.6 SpostaPoesiaInRaccolta	85
6.2.7 ModificaRaccolta	86
6.2.8 commentaPoesia	87
6.2.9 GeneraReport	87
6.2.10 RicercaPoesiaPerTag	88
6.2.11 RicercaPoesiaPerTesto	88
6.2.12 RicercaPoesiaPerTitolo	89

I. Specifiche Informali

Si desidera sviluppare un sistema software per la gestione di una piattaforma sociale dedicata alla scrittura e condivisione di poesie brevi. La piattaforma consentirà agli utenti di pubblicare le proprie poesie, organizzare raccolte tematiche e interagire con i contenuti altrui tramite apprezzamenti e commenti.

Descrizione del Sistema:

Il sistema permette la registrazione di utenti tramite e-mail, nome, cognome e password. Ogni utente dispone di un profilo personale modificabile, che include i propri dati anagrafici, una breve biografia ed eventualmente un'immagine del profilo (identificata da un file immagine). Una volta autenticato, l'utente può pubblicare, mediante apposita interfaccia grafica, nuove poesie brevi (massimo 500 caratteri), assegnando a ciascuna un titolo, uno o più tag tematici (es. "amore", "notte", "solitudine") e scegliendo se includerla in una propria raccolta già esistente, oppure in una nuova raccolta da creare al momento. L'utente può inoltre scegliere di visualizzare l'elenco delle proprie raccolte e l'elenco di tutte le poesie pubblicate.

Ogni raccolta è identificata da un titolo e da una descrizione, ed è composta da un insieme di poesie dello stesso autore. Le poesie possono essere rese pubbliche o private. Le poesie pubbliche sono visibili nel feed degli utenti.

Una volta autenticato, ogni autore può scegliere di visualizzare nel proprio feed principale un elenco delle nuove poesie pubblicate dagli altri autori. Il feed personale mostra al massimo cinque poesie, corrispondenti alle ultime pubblicazioni effettuate da altri autori, ordinate in ordine cronologico decrescente.

Gli utenti possono interagire con le poesie altrui esprimendo il proprio apprezzamento (un "cuore") ed eventualmente lasciando un commento testuale. Ogni commento è identificato da autore, testo e data.

Quando una poesia viene visualizzata, sono mostrati anche il numero totale di cuori ricevuti e l'elenco degli ultimi tre commenti ricevuti.

Ogni utente ha accesso a una sezione "Statistiche" che mostra il numero totale di cuori ricevuti dalle sue poesie, il numero di commenti ricevuti sulle proprie poesie e la poesia più apprezzata (ossia che ha ottenuto il maggior numero di cuori).

Gli amministratori della piattaforma (previa registrazione ed autenticazione) possono generare report contenenti dati aggregati come il numero di poesie pubblicate in un determinato intervallo di tempo, gli autori più attivi (che hanno pubblicato il maggior numero di poesie totali) ed eventualmente i tag più utilizzati e le poesie con più interazioni.

La piattaforma deve essere accessibile sia da desktop che da dispositivi mobili, prevedere notifiche in tempo reale (per nuovi commenti o cuori ricevuti), e garantire la protezione dei dati personali, la sicurezza delle informazioni salvate e un sistema di autenticazione robusto.

2. Analisi e specifica dei requisiti

2.1 Analisi nomi-verbi

Il sistema permette la registrazione di utenti tramite e-mail, nome, cognome e password. Ogni utente dispone di un profilo personale modificabile, che include i propri dati anagrafici, una breve biografia ed eventualmente un'immagine del profilo (identificata da un file immagine). Una volta autenticato, l'utente può pubblicare, mediante apposita interfaccia grafica, nuove poesie brevi (massimo 500 caratteri), assegnando a ciascuna un titolo, uno o più tag tematici (es. "amore", "notte", "solitudine") e scegliendo se includerla in una propria raccolta già esistente, oppure in una nuova raccolta da creare al momento. L'utente può inoltre scegliere di visualizzare l'elenco delle proprie raccolte e l'elenco di tutte le poesie pubblicate.

Ogni raccolta è identificata da un titolo e da una descrizione, ed è composta da un insieme di poesie dello stesso autore. Le poesie possono essere rese pubbliche o private. Le poesie pubbliche sono visibili nel feed degli utenti.

Una volta autenticato, ogni autore può scegliere di visualizzare nel proprio feed principale un elenco delle nuove poesie pubblicate dagli altri autori. Il feed personale mostra al massimo cinque poesie, corrispondenti alle ultime pubblicazioni effettuate da altri autori, ordinate in ordine cronologico decrescente.

Gli utenti possono interagire con le poesie altrui esprimendo il proprio apprezzamento (un "cuore") ed eventualmente lasciando un commento testuale. Ogni commento è identificato da autore, testo e data.

Quando una poesia viene visualizzata, sono mostrati anche il numero totale di cuori ricevuti e l'elenco degli ultimi tre commenti ricevuti.

Ogni utente ha accesso a una sezione "Statistiche" che mostra il numero totale di cuori ricevuti dalle sue poesie, il numero di commenti ricevuti sulle proprie poesie e la poesia più apprezzata (ossia che ha ottenuto il maggior numero di cuori).

Gli amministratori della piattaforma (previa registrazione ed autenticazione) possono generare report contenenti dati aggregati come il numero di poesie pubblicate in un determinato intervallo di tempo, gli autori più attivi (che hanno pubblicato il maggior numero di poesie totali) ed eventualmente i tag più utilizzati e le poesie con più interazioni.

La piattaforma deve essere accessibile sia da desktop che da dispositivi mobili, prevedere notifiche in tempo reale (per nuovi commenti o cuori ricevuti), e garantire la protezione dei dati personali, la sicurezza delle informazioni salvate e un sistema di autenticazione robusto.

LEGENDA:

Classe

Attributo

Funzionalità

Classe-Attore

2.2 Revisione dei requisiti

1. Il sistema deve offrire all'Utente una funzionalità per registrarsi.
2. Il sistema deve offrire all'Utente autenticato una funzionalità per modificare il proprio profilo personale.
3. Di ogni Profilo si vuole memorizzare i dati anagrafici, la biografia e un'immagine del profilo.
4. Il sistema deve permettere all'utente di autenticarsi.
5. Il sistema deve offrire all'Utente autenticato una funzionalità per pubblicare una nuova poesia breve.
6. Di ogni Poesia si vuole memorizzare il titolo, il testo (massimo 500 caratteri), lo stato di pubblicazione (pubblica o privata), i tag associati, la raccolta di appartenenza, la data di pubblicazione e il numero di cuori ricevuti.
7. Il sistema deve permettere all'Utente di associare la poesia a una raccolta esistente.
8. Il sistema deve offrire all'Utente una funzionalità per creare una nuova raccolta al momento della pubblicazione della poesia.
9. Di ogni Raccolta si vuole memorizzare il titolo, la descrizione e l'elenco delle poesie associate.
10. Il sistema deve offrire all'Utente una funzionalità per gestire l'elenco delle proprie raccolte.
11. Il sistema deve offrire all'Utente una funzionalità per visualizzare l'elenco delle proprie poesie pubblicate.
12. Il sistema deve permettere di rendere una poesia pubblica o privata.
13. Il sistema deve offrire all'Autore una funzionalità per visualizzare nel proprio feed principale le nuove poesie pubblicate dagli Autori.
14. Il sistema deve mostrare nel feed personale al massimo cinque poesie (pubbliche), corrispondenti alle ultime pubblicazioni degli Autori, ordinate in ordine cronologico decrescente.
15. Il sistema deve offrire agli Utenti una funzionalità per esprimere un apprezzamento (cuore) per una poesia altrui.
16. Il sistema deve offrire agli Utenti una funzionalità per commentare una poesia altrui inserendo un commento testuale.
17. Di ogni Commento si vuole memorizzare l'autore, il testo e la data del commento.
18. Il Sistema deve mostrare, durante la visualizzazione di una poesia, il numero totale di cuori ricevuti e l'elenco degli ultimi tre commenti ricevuti.

19. Il sistema deve offrire all'Utente una sezione "Statistiche" per visualizzare il numero totale di cuori ricevuti dalle proprie poesie.
20. Il sistema deve mostrare nella sezione "Statistiche" il numero totale di commenti ricevuti sulle proprie poesie.
21. Il sistema deve mostrare nella sezione "Statistiche" la poesia che ha ricevuto il maggior numero di cuori.
22. Il sistema deve offrire all'Amministratore una funzionalità per generare report contenenti dati aggregati.
23. Il sistema deve offrire all'Utente una funzionalità per ricercare poesie (tag, testo e titolo).
24. Il sistema deve offrire all'Utente una funzionalità per modificare una poesia.
25. Ogni Report deve contenere informazioni riguardanti il numero di poesie pubblicate in un intervallo di tempo, gli Autori più attivi, i tag più utilizzati e le poesie con più interazioni.
26. La piattaforma deve essere accessibile sia da desktop che da dispositivi mobili.
27. Il sistema deve inviare notifiche in tempo reale all'Utente per ogni nuovo commento ricevuto su una sua poesia.
28. Il sistema deve inviare notifiche in tempo reale all'Utente per ogni nuovo cuore ricevuto su una sua poesia.
29. Il sistema deve garantire la protezione dei dati personali.
30. Il sistema deve garantire la sicurezza delle informazioni salvate.
31. Il sistema deve implementare un sistema di autenticazione robusto.

2.3 Glossario dei termini

Termino	Descrizione	Sinonimi
Utente	È l'utilizzatore della piattaforma. Può interagire con il sistema previa registrazione e autenticazione.	Autore, Utente registrato
Profilo personale	Insieme di informazioni e contenuti relativi ad un utente registrato.	Profilo
Poesia breve	Elemento informativo centrale del sistema.	Poesia
Raccolta	Insieme di poesie brevi appartenenti allo stesso autore, identificato da un nome e una descrizione.	
Feed	Sezione personale in cui l'utente autenticato visualizza una selezione cronologicamente ordinata di poesie pubbliche pubblicate dagli autori. Mostra un massimo di cinque poesie recenti.	Feed personale, Feed principale
Statistiche	Sezione del profilo utente che fornisce informazioni aggregate sull'attività dell'utente nella piattaforma.	
Amministratore della piattaforma	Utenti con privilegi associati alla gestione della piattaforma.	Amministratore
Report	Insieme di informazioni aggregate e approfondimenti sull'attività della piattaforma, generati dagli amministratori per fini di analisi e monitoraggio.	

2.4 Classificazione dei requisiti

2.4.1 Requisiti funzionali

ID	Requisito	Origine (n. frase dei requisiti revisionati)
RF01	Il sistema deve offrire all'Utente una funzionalità per registrarsi.	1
RF02	Il sistema deve offrire all'Utente autenticato una funzionalità per modificare il proprio profilo personale.	2
RF03	Il sistema permette all'utente di autenticarsi	4
RF04	Il sistema deve offrire all'Utente autenticato una funzionalità per pubblicare una nuova poesia breve.	5
RF05	Il sistema deve permettere all'Utente di associare la poesia a una raccolta esistente.	7
RF06	Il sistema deve offrire all'Utente una funzionalità per creare una nuova raccolta al momento della pubblicazione della poesia.	8
RF07	Il sistema deve offrire all'Utente una funzionalità per visualizzare e modificare le proprie raccolte.	10
RF08	Il sistema deve offrire all'Utente una funzionalità per visualizzare l'elenco delle proprie poesie pubblicate.	11
RF09	Il sistema deve permettere di rendere una poesia pubblica o privata.	12
RF10	Il sistema deve offrire all'Autore una funzionalità per visualizzare nel proprio feed principale le nuove poesie pubblicate dagli Autori.	13, 14

RF11	Il sistema deve offrire agli Utenti una funzionalità per esprimere un apprezzamento (cuore) per una poesia altrui.	15
RF12	Il sistema deve offrire agli Utenti una funzionalità per commentare una poesia altrui inserendo un commento testuale.	16
RF13	Il Sistema deve mostrare, durante la visualizzazione di una poesia, il numero totale di cuori ricevuti e l'elenco degli ultimi tre commenti ricevuti.	18
RF14	Il sistema deve offrire all'Utente una sezione "Statistiche" per visualizzare il numero totale di cuori ricevuti dalle proprie poesie.	19
RF15	Il sistema deve mostrare nella sezione "Statistiche" il numero totale di commenti ricevuti sulle proprie poesie.	20
RF16	Il sistema deve mostrare nella sezione "Statistiche" la poesia che ha ricevuto il maggior numero di cuori.	21
RF17	Il sistema deve offrire all'Amministratore una funzionalità per generare report contenenti dati aggregati.	22
RF18	Il sistema deve offrire all'Utente una funzionalità per ricercare poesie per tag, per testo o per titolo.	23

2.4.2 Requisiti sui dati

ID	Requisito	Origine (n. frase dei requisiti revisionati)
RD₀₁	Di ogni Profilo si vuole memorizzare i dati anagrafici, la biografia e un'immagine del profilo	3
RD₀₂	Di ogni Poesia si vuole memorizzare il titolo, il testo (massimo 500 caratteri), lo stato di pubblicazione (pubblica o privata), i tag associati, la raccolta di appartenenza, la data di pubblicazione e il numero di cuori ricevuti	6
RD03	Di ogni Raccolta si vuole memorizzare il titolo, la descrizione e l'elenco delle poesie associate	9
RD04	Di ogni Commento si vuole memorizzare l'autore, il testo e la data del commento.	17
RD05	Ogni Report deve contenere informazioni riguardanti il numero di poesie pubblicate in un intervallo di tempo, gli Autori più attivi, i tag più utilizzati e le poesie con più interazioni.	25

2.4.3 Vincoli / Altri requisiti

ID	Requisito	Origine (n. frase dei requisiti revisionati)
V₀₁	Il sistema deve mostrare le poesie pubbliche nel feed degli Utenti	13
V₀₂	Il sistema deve mostrare nel feed personale al massimo cinque poesie, corrispondenti alle ultime pubblicazioni degli Autori, ordinate in ordine cronologico decrescente.	14

RDS	La piattaforma deve essere accessibile sia da desktop che da dispositivi mobili.	26
RDS	Il sistema deve inviare notifiche in tempo reale all'Utente per ogni nuovo commento ricevuto su una sua poesia.	27
RDS	Il sistema deve inviare notifiche in tempo reale all'Utente per ogni nuovo cuore ricevuto su una sua poesia.	28
RDS	Il sistema deve garantire la protezione dei dati personali.	29
RDS	Il sistema deve garantire la sicurezza delle informazioni salvate	30
RDS	Il sistema deve implementare un sistema di autenticazione robusto.	31

2.5 Modellazione dei casi d'uso

2.5.1 Attori e casi d'uso

Attori Primari:

- Utente
- UtenteNonRegistrato
- Amministratore (ext. Utente)

Attori Secondari

- ServizioMessagistica

Casi d'uso:

- UC1: Registrazione
- UC2: Autenticazione
- UC3: GestioneProfiloPersonale
- UC4: PubblicazionePoesia
- UC5: GestioneRaccolte
- UC6: VisualizzazionePoesia
- UC7: VisualizzazioneFeed
- UC8: CommentoPoesia
- UC9: ApprezzamentoPoesia
- UC10: GeneraReport
- UC11: VisualizzazioneStatistiche
- UC12: RicercaPoesie

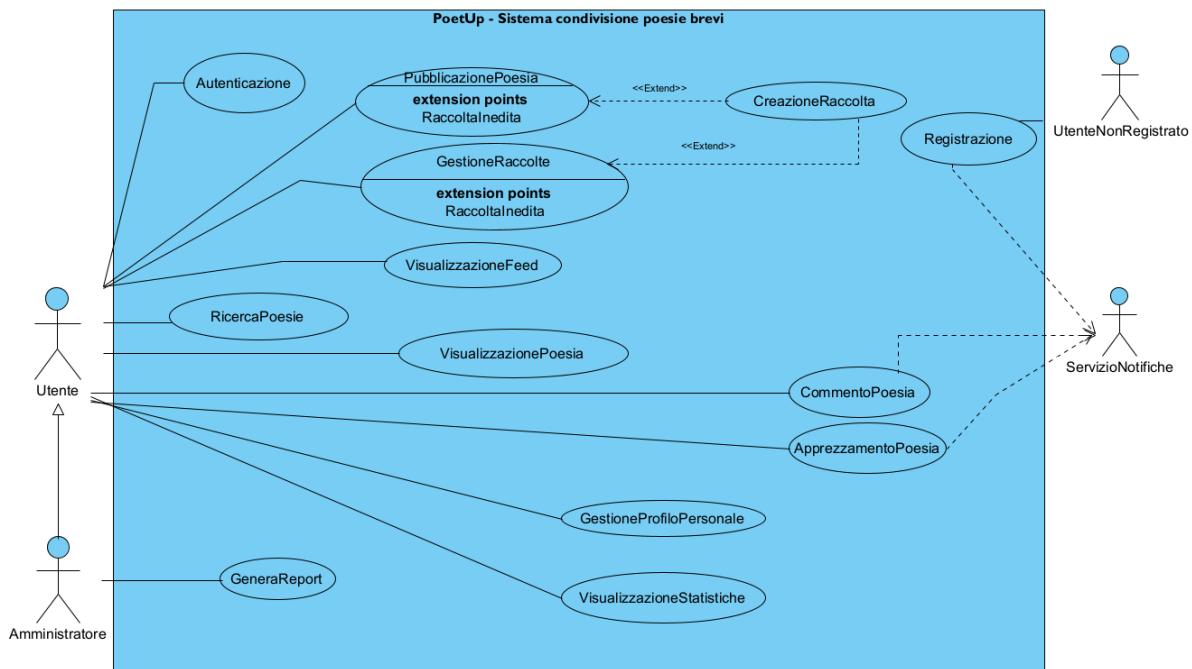
Casi d'uso di estensione:

- UC13: CreazioneRaccolta

Caso d'uso	Attori Primari	Attori Secondari	Incl. / Ext.	Requisiti corrispondenti
UC1: Registrazione	UtenteNonRegistrato	ServizioNotifiche	-	RF01
UC2: Autenticazione	Utente	-	-	RF03
UC3: GestioneProfiloPersonale	Utente	-	-	RF02
UC4: PubblicazionePoesia	Utente	-	È esteso da: CreazioneRaccolta	RF04, RF05, RF09
UC5: GestioneRaccolte	Utente	-	È esteso da: CreazioneRaccolta ModificaPoesia	RF07
UC6: VisualizzazionePoesia	Utente			RF13
UC7: VisualizzazioneFeed	Utente	-	-	RF10
UC8: CommentoPoesia	Utente	ServizioNotifiche	-	RF12

UC9: ApprezzamentoPoesia	Utente	-	-	RF11
UC10: GeneraReport	Amministratore	-	-	RF17
UC11: VisualizzazioneStatistiche	Utente	-	-	RF14, RF15, RF16
UC12: RicercaPoesie	Utente	-	-	RF18
UC13: CreazioneRaccolta	Utente	-	Estende: PubblicazionePoesia, GestioneRaccolte	RF06

2.5.2 Diagramma dei casi d'uso



2.5.3 Scenari

Caso d'uso	UCI: Registrazione
Attore primario	Utente non registrato
Attore secondario	ServizioNotifiche
Descrizione	Un Utente non registrato si registra alla piattaforma
Pre-condizioni	-
Sequenza di eventi principale	<ol style="list-style-type: none"> Il caso d'uso inizia quando l'Utente non registrato richiede la registrazione. L'Utente non registrato inserisce indirizzo e-mail, password e username. Il sistema controlla che i dati inseriti siano validi sintatticamente. Il sistema mostra un messaggio per la registrazione avvenuta con successo. Il sistema aggiorna le informazioni relative al profilo utente sul database. Il sistema torna alla scheda del login.
Post-condizioni	Il sistema ha memorizzato i dati inseriti nell'elenco Utenti
Casi d'uso correlati	-
Sequenza di eventi alternativi	<p>Al punto 2 se i dati inseriti non sono validi, ovvero:</p> <ol style="list-style-type: none"> La password non contiene almeno un numero o carattere speciale o non è compresa tra gli 8 e i 32 caratteri.

	<p>b. L'indirizzo email non è in un formato riconosciuto. c. L'indirizzo e-mail inserito è già registrato. d. Uno dei campi non è stato inserito.</p> <p>allora:</p> <ul style="list-style-type: none"> a. Il sistema visualizza un messaggio di errore specificando quale controllo è fallito. b. Il sistema ripropone il form all'Utente non registrato. c. Torna al punto 2.
Caso d'uso	UC2: Autenticazione
Attore primario	Utente
Attore secondario	-
Descrizione	Il caso d'uso descrive il processo attraverso il quale un utente già registrato accede al sistema tramite le proprie credenziali.
Pre-condizioni	-
Sequenza di eventi principale	<ol style="list-style-type: none"> 1. L'utente accede alla pagina di login del sistema. 2. Il sistema presenta il modulo di autenticazione. 3. L'utente inserisce le proprie credenziali (email e password). 4. L'utente preme il pulsante “Accedi”. 5. Il sistema verifica le credenziali inserite. 6. if Le credenziali sono corrette. <ul style="list-style-type: none"> a. Il sistema autentica l'utente e lo reindirizza alla homepage
Post-condizioni	L'utente è autenticato e ha accesso alle funzionalità riservate del sistema.
Casi d'uso correlati	-
Sequenza di eventi alternativi	<p>Al punto 6, se le credenziali non sono corrette:</p> <ul style="list-style-type: none"> a. Il sistema mostra un messaggio di errore: “Email o password non corretti.” b. Ritorna al punto 2.
Caso d'uso:	UC3: GestioneProfiloPersonale
Attore primario	Utente
Attore secondario	-

Descrizione	L'utente accede alla sezione del proprio profilo personale dove può visualizzare e modificare le informazioni relative al profilo personale.
Pre-Condizioni	-
Sequenza di eventi principale	<ol style="list-style-type: none"> 1. Il caso d'uso inizia quando l'utente entra nella sezione "Profilo personale" tramite menù. 2. Il sistema mostra i campi vuoti del profilo (nome, cognome, biografia, immagine del profilo e data di nascita) eccetto il nickname che è stato impostato dalla registrazione ed è immodificabile. 3. L'utente procede a modificare i dati 4. L'utente seleziona il pulsante "Salva Modifiche" 5. Il sistema valida i dati presenti nei campi. 6. if i dati sono validi: <ol style="list-style-type: none"> 6.1. Il sistema aggiorna il profilo dell'utente nel database. 6.2. Il sistema mostra un messaggio di conferma: "Profilo aggiornato con successo."
Post-Condizioni	Il profilo personale dell'utente è stato aggiornato con i nuovi dati.
Casi d'uso correlati	-
Sequenza di eventi alternativi	<p>Al punto 6 se i dati inseriti non sono validi, ovvero:</p> <ol style="list-style-type: none"> a. Il formato dell'immagine non è supportato. b. Uno dei campi è stato lasciato vuoto. <p>allora:</p> <ol style="list-style-type: none"> a. Il sistema visualizza un messaggio di errore b. Torna al punto 3.
Caso d'uso	UC4: PubblicazionePoesia
Attore primario	Utente
Attore secondario	-
Descrizione	L'utente pubblica una nuova poesia breve sulla piattaforma.
Pre-condizioni	
Sequenza di eventi principale	<ol style="list-style-type: none"> 1. L'utente accede alla sezione "Nuova Poesia". 2. Il sistema presenta il form per titolo, testo, tag, visibilità e raccolta.

	<p>3. L'utente inserisce il titolo, il testo (max 500 caratteri), la raccolta, assegna un tag e imposta la visibilità della poesia.</p> <p>4. Il sistema verifica che la lunghezza della poesia e del titolo sia valida.</p> <p>5. if le verifiche sono andate a buon fine</p> <p>5.1. if la raccolta selezionata non esiste</p> <p>5.1.1. <<esteso>> <i>CreazioneRaccolta</i></p> <p>5.2. L'utente invia il modulo</p> <p>5.3. Il sistema memorizza la poesia e notifica conferma.</p>
Post-condizioni	La poesia viene memorizzata nel database.
Casi d'uso correlati	<i>CreazioneRaccolta</i>
Sequenza di eventi alternativi	Al punto 5, se il testo della poesia contiene meno di 1 carattere o più di 500 , o il titolo ha una dimensione minore di 1 parola , il sistema non accetta l'invio e:
	a. Mostra un messaggio di errore specificando il limite richiesto. b. Ripropone il modulo già compilato, consentendo la correzione del testo. c. Torna al punto 4
Caso d'uso	UC5: GestioneRaccolte
Attore primario	Utente
Attore secondario	-
Descrizione	Il caso d'uso descrive le funzionalità per la visualizzazione, modifica e gestione delle raccolte personali di poesie da parte dell'Utente.
Pre-condizioni	-
Sequenza di eventi principale	<p>1. L'Utente accede alla sezione “Raccolte” tramite il menù dell'interfaccia.</p> <p>2. Il Sistema recupera e visualizza l'elenco delle raccolte associate all'utente.</p> <p>3. if L'utente clicca sul ‘+’</p> <p>3.1. <<esteso>> <i>CreazioneRaccolta</i></p> <p>4. else if L'Utente seleziona una raccolta</p> <p>4.1. Il Sistema mostra l'elenco delle poesie contenute nella raccolta selezionata.</p> <p>4.2. Per ogni poesia, il sistema visualizza una “X”.</p> <p>4.3. if L'utente clicca sulla “X”</p> <p>4.3.1. L'utente sceglie se spostare la poesia o eliminarla.</p> <p>4.3.2. if l'utente sceglie “sposta”:</p> <p>4.3.2.1. l'Utente fornisce tramite interfaccia il nome della raccolta destinazione</p> <p>4.3.3. else if l'utente sceglie “elimina”:</p> <p>4.3.3.1. La poesia viene rimossa dal database</p> <p>4.3.4. Il Sistema elimina la poesia dalla raccolta e aggiorna le informazioni sul database</p> <p>4.3.5. Il Sistema torna al punto 4.1.</p> <p>4.4. Il Sistema visualizza, accanto al titolo della raccolta due opzioni: Modifica e Elimina.</p> <p>4.5. if L'Utente clicca sull'icona Modifica.</p>

	<p>4.5.1. Il Sistema mostra le opzioni: “Modifica titolo”, “Modifica descrizione”.</p> <p>4.5.2. L’Utente seleziona una delle opzioni e modifica le informazioni.</p> <p>4.5.3. Il Sistema aggiorna le informazioni della raccolta nel database.</p> <p>4.6. elif L’Utente clicca sull’icona Elimina.</p> <p>4.6.1. Il Sistema mostra un messaggio per la conferma dell’eliminazione</p> <p>4.6.2. if L’Utente clicca su si.</p> <p>4.6.2.1. La raccolta viene rimossa dal database</p> <p>4.6.3. elif L’Utente torna al punto 2.</p>
Post-condizioni	Le informazioni sulle raccolte e poesie associate dell’utente risultano aggiornate.
Casi d’uso correlati	-
Sequenza di eventi alternativi	Al punto 2 se non sono presenti raccolte, il sistema mostra un messaggio: “Nessuna raccolta disponibile”.
Caso d’uso:	UC6: VisualizzazionePoesia
Attore primario	Utente
Attore secondario	-
Descrizione	Il sistema mostra all’utente una pagina di visualizzazione di una poesia.
Pre-Condizioni	-
Sequenza di eventi principale	<ol style="list-style-type: none"> 1. L’utente ha selezionato una poesia da una schermata di selezione 2. Il sistema mostra all’utente un’interfaccia contenente il titolo della poesia, l’autore, il testo, il tag, un cuore per gli apprezzamenti, gli ultimi tre commenti, il numero totale di apprezzamenti ricevuti, lo stato, un pulsante “X” per poter eliminare o spostare la poesia in un’altra Raccolta e un pulsante chiudi per chiudere la schermata della Poesia.
Post-Condizioni	-

Casi d'uso correlati	-
Sequenza di eventi alternativi	-
Caso d'uso:	UC7: VisualizzazioneFeed
Attore primario	Utente
Attore secondario	-
Descrizione	Il sistema mostra un elenco di massimo 5 poesie pubbliche pubblicate da altri utenti ordinate dalla più recente alla meno recente.
Pre-Condizioni	-
Sequenza di eventi principale	<ol style="list-style-type: none"> 1. L'utente autenticato accede alla sezione "Home". 2. Il sistema interroga il database per recuperare le poesie pubbliche pubblicate da altri autori. 3. Il sistema ordina le poesie per data di pubblicazione, in ordine decrescente (dalla più recente). 4. Il sistema seleziona le prime cinque poesie dall'elenco ordinato. 5. Il sistema mostra nel feed dell'utente (max. 5 poesie): <ul style="list-style-type: none"> a. -Titolo della poesia b. -Autore c. -data di pubblicazione d. -numero apprezzamenti
Post-Condizioni	L'utente ha visualizzato correttamente un feed aggiornato con le ultime poesie pubbliche degli altri utenti.
Casi d'uso correlati	-

Sequenza di eventi alternativi	Al punto 5 se non vi sono poesia pubblicate: a. Il sistema mostra un messaggio: “ <i>Nessuna poesia pubblica disponibile al momento</i> ”.
Caso d'uso	UC8: CommentoPoesia
Attore primario	Utente
Attore secondario	ServizioNotifiche
Descrizione	L'Utente pubblica un commento relativo ad una poesia pubblica.
Pre-condizioni	L'Utente sta visualizzando la pagina di dettaglio di una poesia.
Sequenza di eventi principale	<ol style="list-style-type: none"> 1. L'Utente seleziona “Commenta” durante la visualizzazione della poesia. 2. Il sistema apre il campo di inserimento testo. 3. L'Utente scrive il commento. 4. L'Utente conferma l'invio. 5. Il sistema valida la lunghezza del testo. 6. if la verifica ha esito positivo <ol style="list-style-type: none"> 6.1. Il sistema memorizza il commento con testo, autore e data. 6.2. Il sistema notifica all'autore della poesia la ricezione di un commento.
Post-condizioni	Il commento viene memorizzato nel database
Casi d'uso correlati	-
Sequenza di eventi alternativi	Al punto 6, se il testo del commento ha una lunghezza maggiori di 255 caratteri o è una stringa vuota : <ol style="list-style-type: none"> a. Il sistema mostra un errore e richiede la modifica del commento b. Torna al punto 3
Caso d'uso	UC9: ApprezzamentoPoesia
Attore primario	Utente
Attore secondario	ServizioNotifiche
Descrizione	L'Utente pubblica un apprezzamento relativo ad una poesia pubblica.
Pre-condizioni	L'Utente sta visualizzando la pagina di dettaglio di una poesia.
Sequenza di eventi principale	<ol style="list-style-type: none"> 1. L'Utente clicca sull'icona cuore. 2. if L'apprezzamento non è già presente (icona cuore vuota): <ol style="list-style-type: none"> 2.1. Il Sistema registra l'apprezzamento nel database. 2.2. Il Sistema aggiorna il contatore dei cuori (+1). 2.3. Il Sistema invia una notifica all'autore della poesia. 2.4. Il Sistema aggiorna l'interfaccia utente (cuore in stato attivo)
Post-condizioni	Numero di apprezzamenti dell'autore della poesia aggiornato

Casi d'uso correlati	-
Sequenza di eventi alternativi	<p>Al punto 2, se l'apprezzamento è già presente (icona cuore pieno):</p> <ul style="list-style-type: none"> a. Il Sistema rimuove l'apprezzamento dal database. b. Il Sistema aggiorna il contatore dei cuori (-1). c. Il Sistema aggiorna l'interfaccia utente (cuore in stato inattivo).
Caso d'uso	UC10: GeneraReport
Attore primario	Amministratore
Attore secondario	-
Descrizione	L'Amministratore genera un report su dati aggregati sulle attività della piattaforma.
Pre-condizioni	L'Amministratore ha effettuato l'autenticazione
Sequenza di eventi principale	<ol style="list-style-type: none"> 1. L'Amministratore accede alla sezione "Report". 2. Il sistema mostra l'interfaccia per la generazione di un nuovo report. 3. L'Amministratore specifica un intervallo di tempo. 4. L'Amministratore seleziona "Genera Report". 5. Il sistema elabora i seguenti dati aggregati: <ul style="list-style-type: none"> a. – numero di poesie pubblicate b. – autori più attivi c. – tag più utilizzati d. – poesie con più interazioni 6. Il sistema mostra all'Amministratore il report generato.
Post-condizioni	Il report viene memorizzato nel database.
Casi d'uso correlati	-
Caso d'uso	UC11: VisualizzazioneStatistiche
Attore primario	Utente
Attore secondario	-
Descrizione	Un Utente visualizza le proprie statistiche.
Pre-condizioni	-
Sequenza di eventi principale	<ol style="list-style-type: none"> 1. L'utente clicca sull'icona per visualizzare le statistiche dal menù. 2. Il sistema calcola il numero di apprezzamenti e il numero di commenti ricevuti. 3. Il sistema mostra all'Utente il numero di apprezzamenti, il numero di commenti ricevuti e la propria poesia più apprezzata.
Post-condizioni	-
Casi d'uso correlati	-

Sequenza di eventi alternativi	Al punto 2 se l'Utente non ha pubblicato alcuna poesia il sistema indica all'Utente che non ci sono statistiche da visualizzare.
C A D P S e	<pre> classDiagram class Poesia { -id -titolo -tag -body -visibilita -contatoreLike -dataPubblicazione -idAutore +like() +commenta() } class Raccolta { -id -titolo -descrizione +addPoesia() +removePoesia() } class ProfiloPersonale { -nome -cognome -immagineProfilo -biografia -nickname -dataNascita } class Commento { -id -testo -data -idAutore -idPoesia } class Apprezzamento { -id -idAutore -idPoesia } class Utente { -email -password +addRaccolta() +removeRaccolta() +calcolaStatistiche() +modificaProfilo() +pubblicazionePoesia() } class Amministratore { +generaReport() } class Report { -id -data -numPoesiePubblicate -leadUtenti -leadTag -leadPoesie -idAutore } class PoetUp { +registrazione() +autenticazione() +ricercaPoesie() +calcoloFeed() } Poesia "1" *-- "0..*" Commento : -commenti Poesia "1" *-- "0..*" Apprezzamento : -apprezzamenti Poesia "1" *-- "0..*" Utente : -poesie ProfiloPersonale "1" *-- "1" Utente : -profilo Amministratore "1" *-- "1" Utente : -generaReport() Utente "1" *-- "1..*" PoetUp : -utenti PoetUp "1" --> "1..*" Report : <<use>> </pre>
	<p>2.2. Ricerca per Testo, 2.3. Ricerca per Titolo. 3. L'Utente sceglie il criterio di ricerca che preferisce e inserisce il contenuto della sua ricerca 4. Il sistema esegue la ricerca tra le poesie pubbliche 5. Il sistema mostra all'Utente l'elenco delle poesie che corrispondono ai criteri inseriti</p>
Post-condizioni	-
Casi d'uso correlati	-
Sequenza di eventi alternativi	Al punto 5 se non esistono poesie che corrispondono ai criteri di ricerca Il sistema informa l'Utente che nessuna poesia corrisponde ai criteri inseriti.

2.6 Diagramma delle classi

TABELLA RESPONSABILITÀ	
registrazione	PoetUp
autenticazione	PoetUp
ricercaPoesia	PoetUp

calcolaFeed	PoetUp
addRaccolta	Utente
removeRaccolta	Utente
calcolaStatistiche	Utente
modificaProfilo	Utente
pubblicazionePoesia	Utente
addPoesia	Raccolta
removePoesia	Raccolta
like	Poesia
commenta	Poesia
generaReport	Amministratore

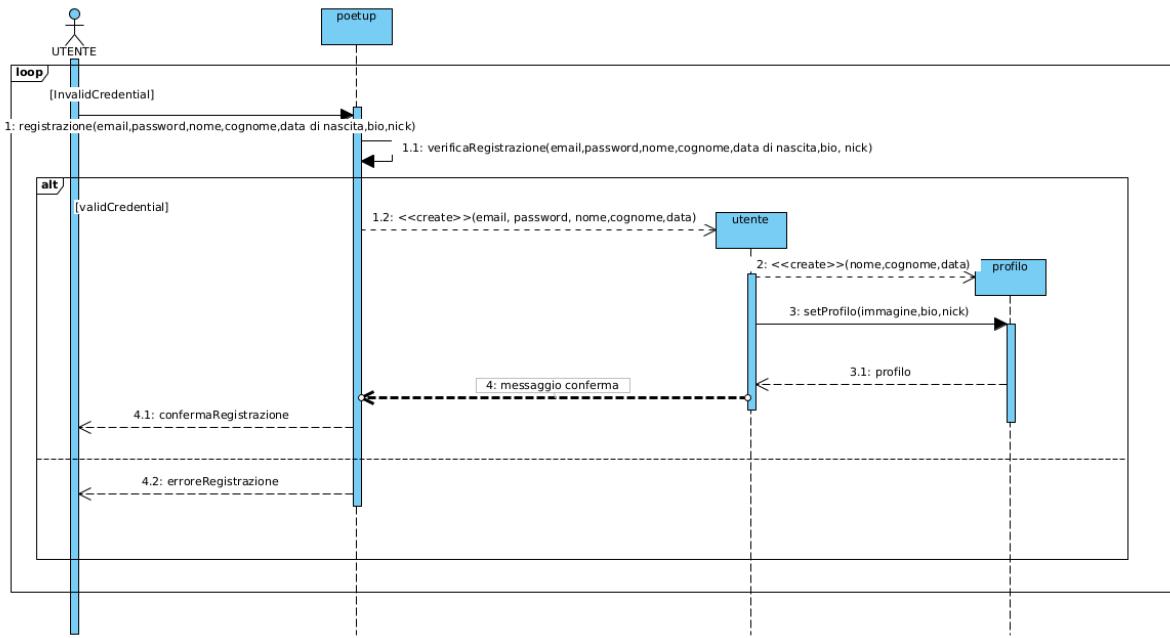
Nel modello di dominio di PoetUp, la creazione degli oggetti è stata affidata seguendo il principio *Creator*, quando opportuno, integrata con il principio di *Information Expert*.

- La classe **Utente** è responsabile della creazione di:
 - **Poesia**, mediante il metodo [pubblicazionePoesia\(\)](#).
 - **Raccolta**, attraverso il metodo [addRaccolta\(\)](#).
 - **ProfiloPersonale**, poiché ciascun utente gestisce il proprio profilo e fornisce i dati necessari alla sua creazione/modifica. L'utente è quindi anche <<Information Expert>> dei suoi dati personali.
- La classe **Poesia** è responsabile della creazione di:
 - **Commento**, attraverso il metodo [commenta\(\)](#), poiché la poesia è il contenitore logico dei commenti e controlla il flusso dell'interazione con l'utente.
 - **Apprezzamento**, tramite [like\(\)](#).
- La classe **Amministratore** è responsabile della creazione di:
 - **Report**, poiché l'amministratore è l'unico attore autorizzato a generare statistiche e riepiloghi sull'attività del sistema. È quindi il naturale <<Creator>> e <<Information Expert>> per l'oggetto **Report**.

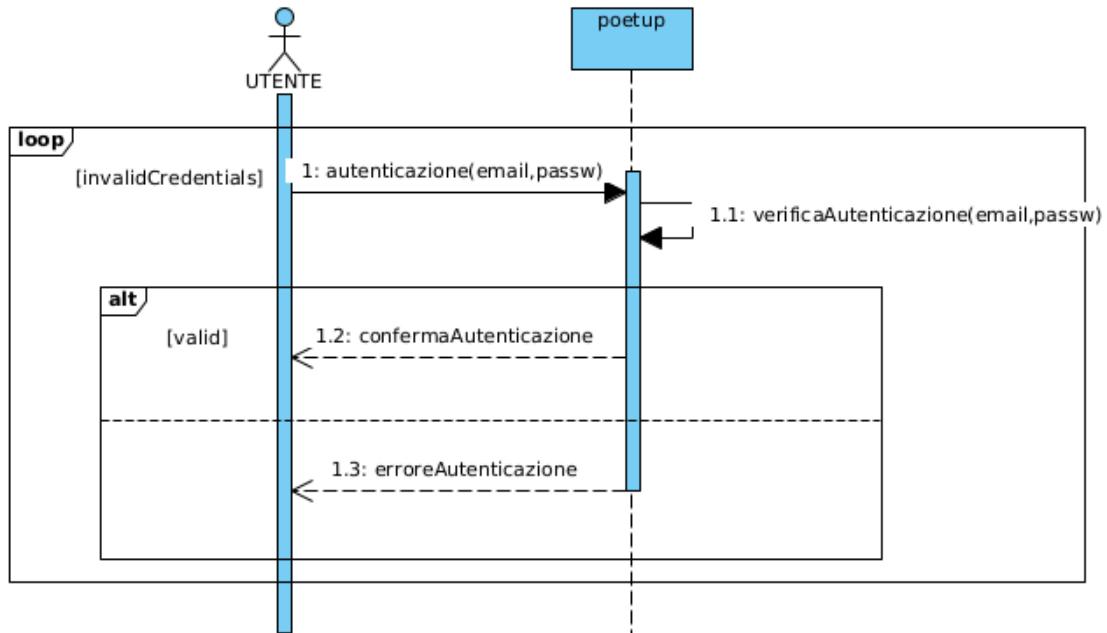
2.7 Diagramma di sequenza

I seguenti diagrammi sono stati realizzati come bozza preliminare per la successiva implementazione del codice.

2.7.1 Registrazione

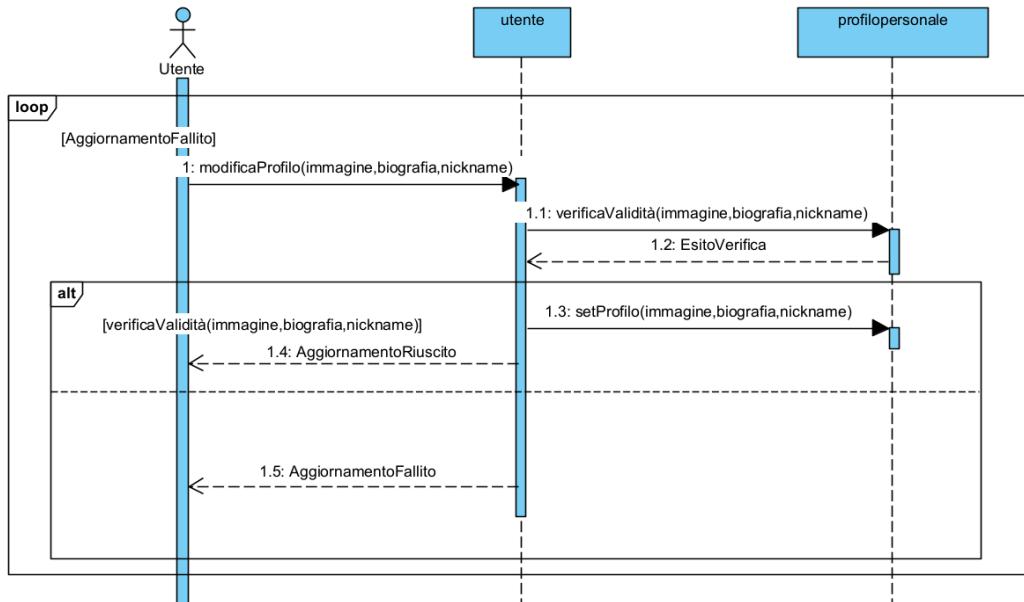


2.7.2 Autenticazione



È stato aggiunto il metodo `verificaAutenticazione(email, passw)` utile a verificare l'effettiva registrazione dell'utente che provi ad autenticarsi.

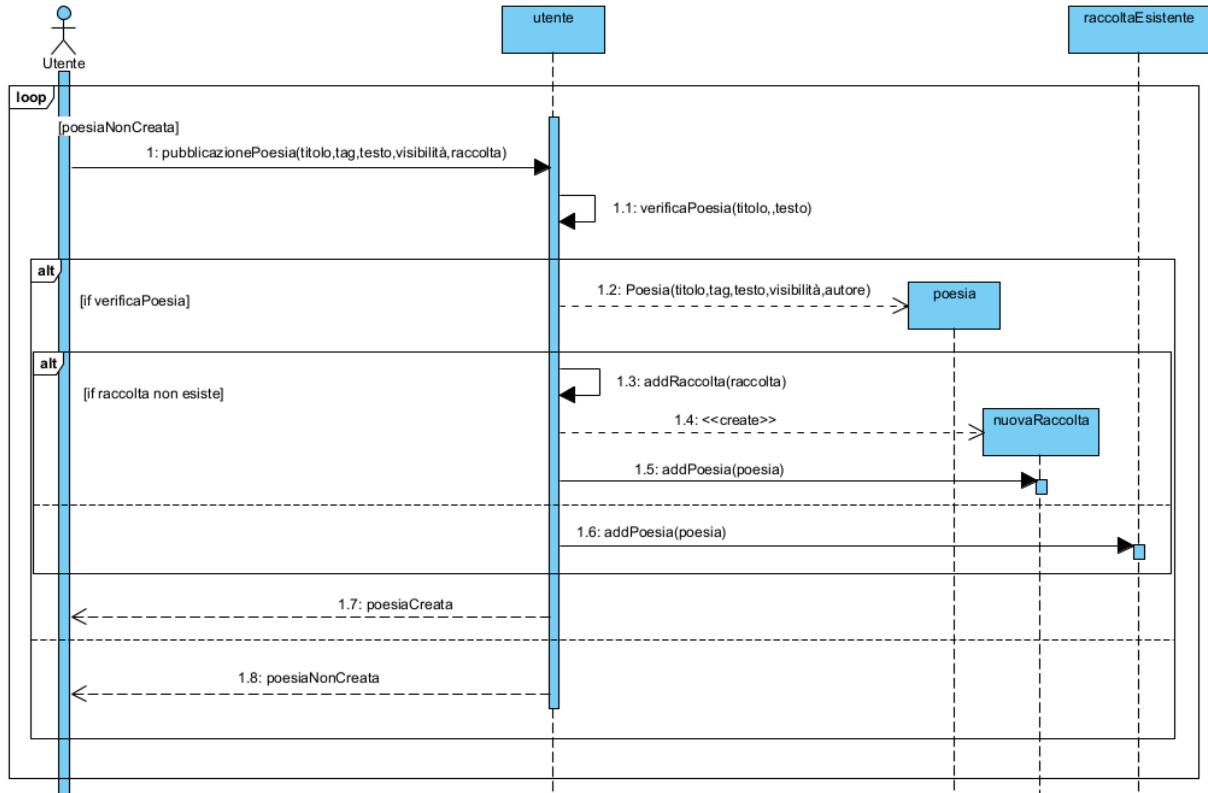
2.7.3 Gestione Profilo Personale



Nel corso dello sviluppo del diagramma è emersa la necessità di un metodo `SetProfilo(nickname, bio, immagine)` per modificare i dati e uno per verificarne la validità, `verificaValidità(nickname, bio,`

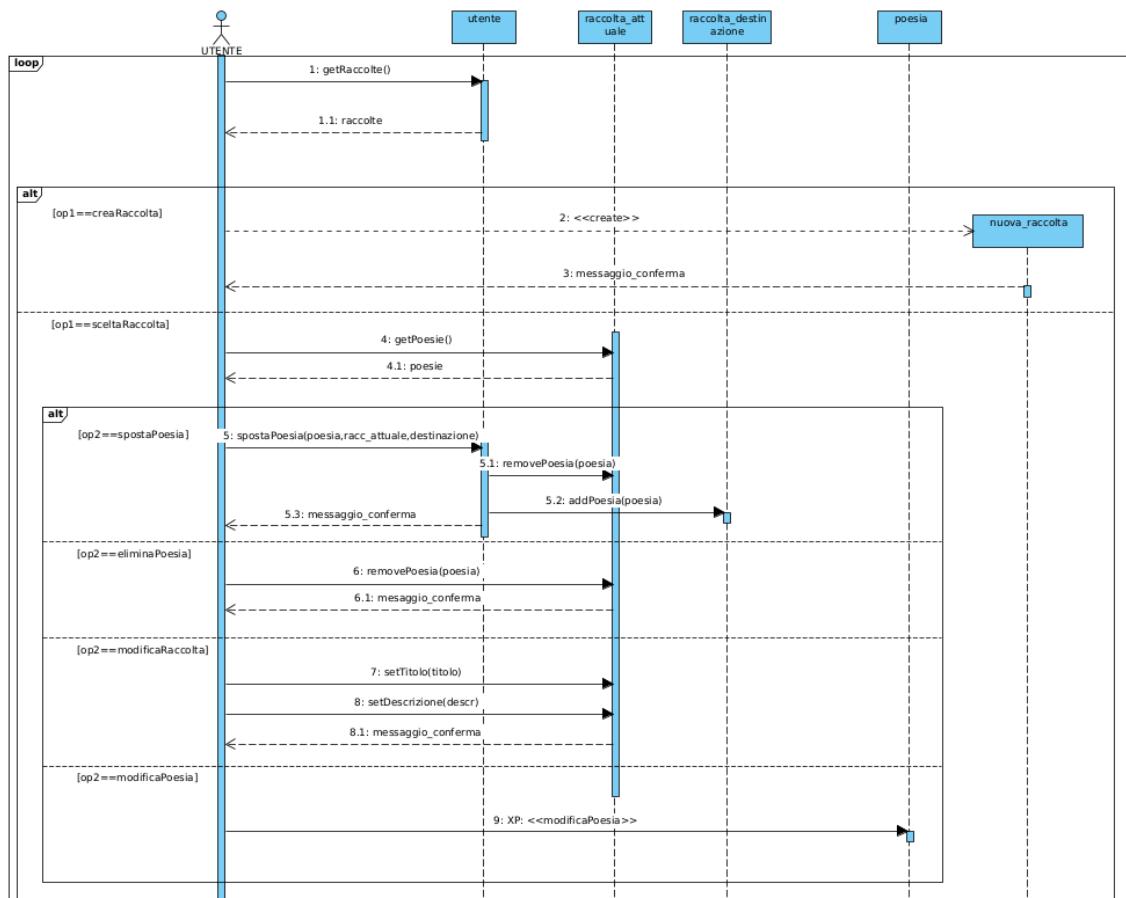
immagine), il quale deve fare dei controlli sul formato dell'immagine, la data di nascita e deve garantire che non venga lasciato vuoto alcuno spazio.

2.7.4 PubblicazionePoesia



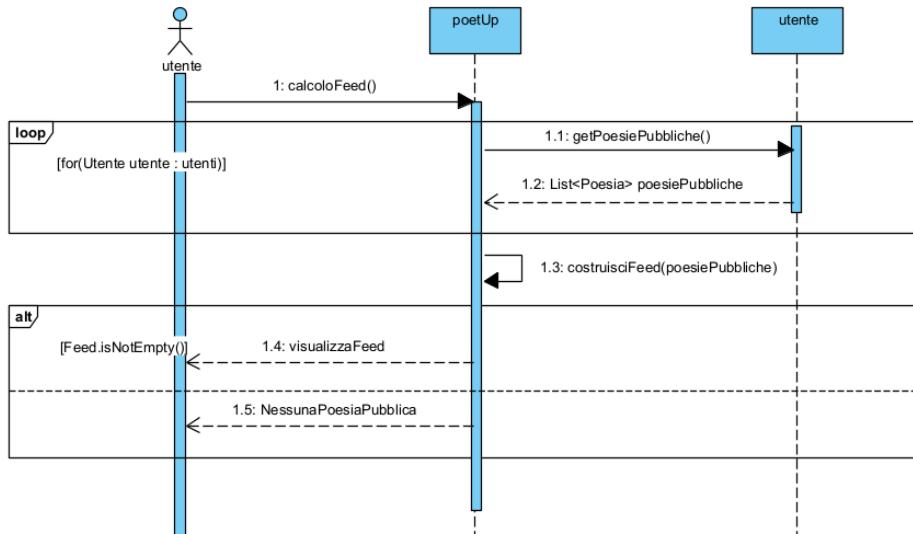
È stato aggiunto un metodo privato di utente per validare la modifica di una poesia.

2.7.5 GestioneRaccolte



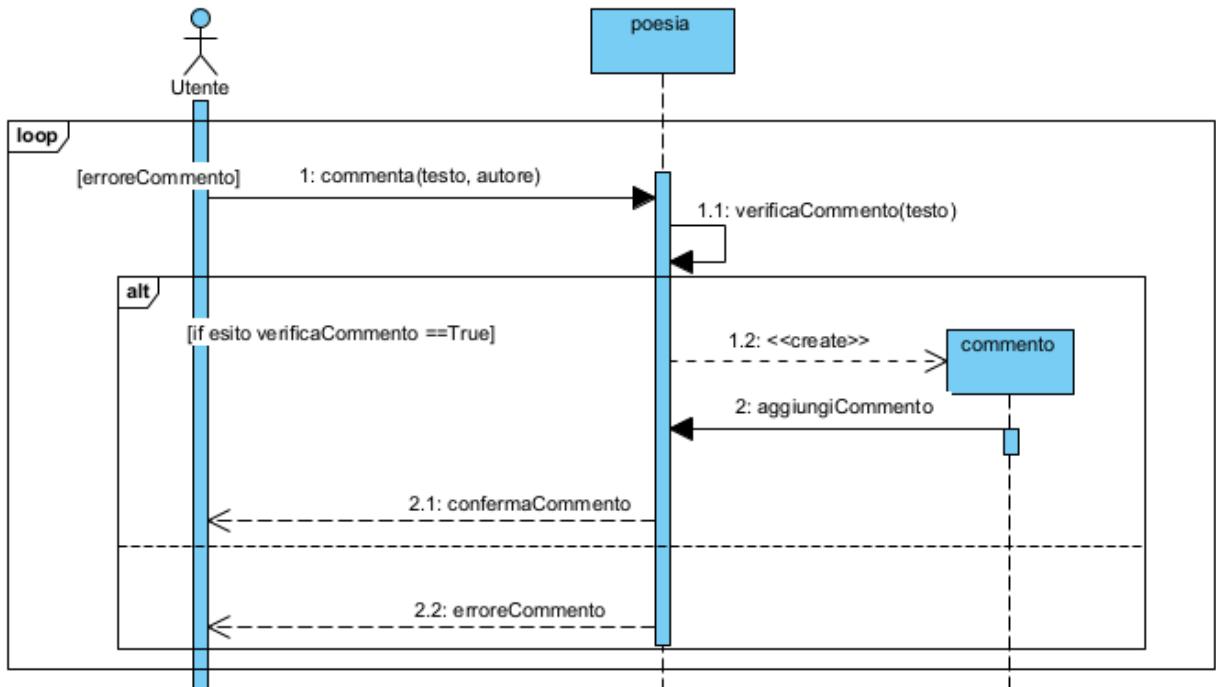
È stato aggiunto il metodo **spostaPoesia** in utente il quale ha visione di tutte le sue raccolte.

2.7.6 VisualizzaFeed



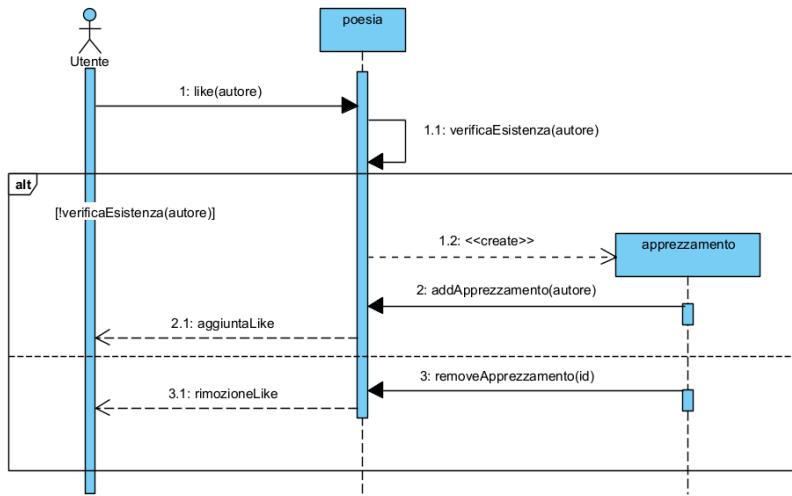
In questo diagramma è emersa la necessità di utilizzare una funzione `costruisciFeed(poesiePubbliche)` che effettui le operazioni necessarie per creare la lista di poesie. Sono stati poi aggiunti `visualizzaFeed()`, che mostrerà titolo e autore per ogni poesia, e `getPoesiePubbliche()` per ottenere una lista di poesie pubbliche.

2.7.7 CommentoPoesia



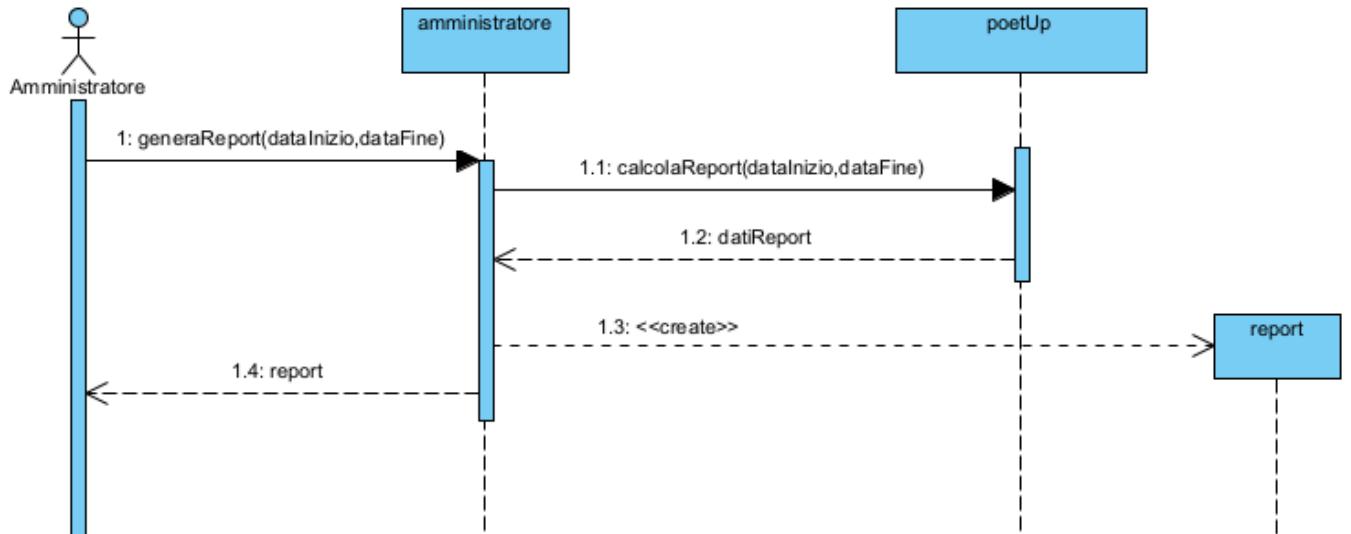
Si è pensato inizialmente di introdurre un metodo privato `verificaCommento(testo)` all'interno della classe Poesia. Questo metodo consente di verificare che il testo del commento rispetti i vincoli stabiliti, ovvero una lunghezza compresa tra 1 e 100 parole.

2.7.8 ApprezzamentoPoesia



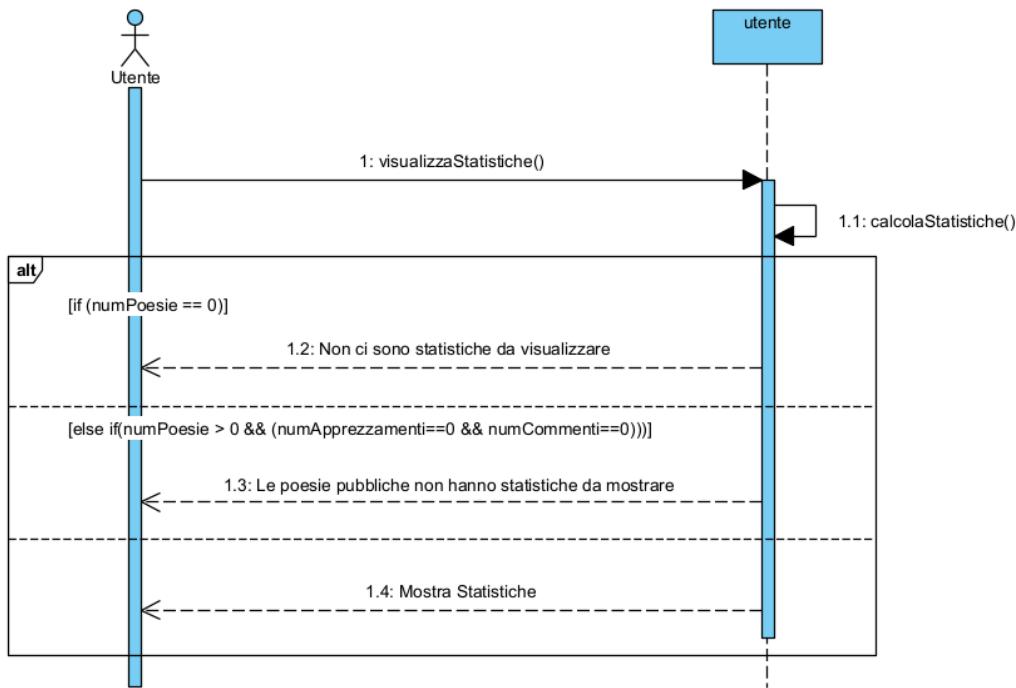
In questo diagramma è emersa la necessità di due metodi, `addApprezzamento()` e `removeApprezzamento()`, utili per l'aggiunta o rimozione di un like. Inoltre, per riconoscere se l'autore avesse già pubblicato un like sulla poesia in questione è stato introdotto il metodo `verificaEsistenza()`.

2.7.9 GeneraReport

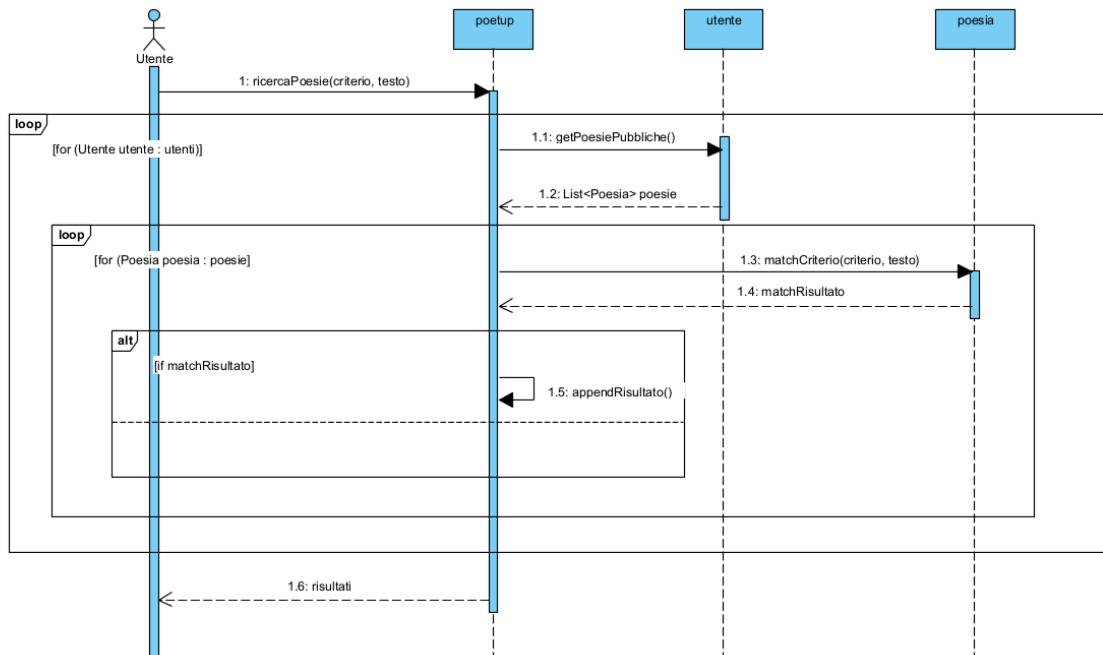


È qui emersa la necessità di aggiungere un metodo `calcolaReport(dataInizio,dataFine)` all'interno della classe PoetUp.

2.7.10 VisualizzaStatistiche

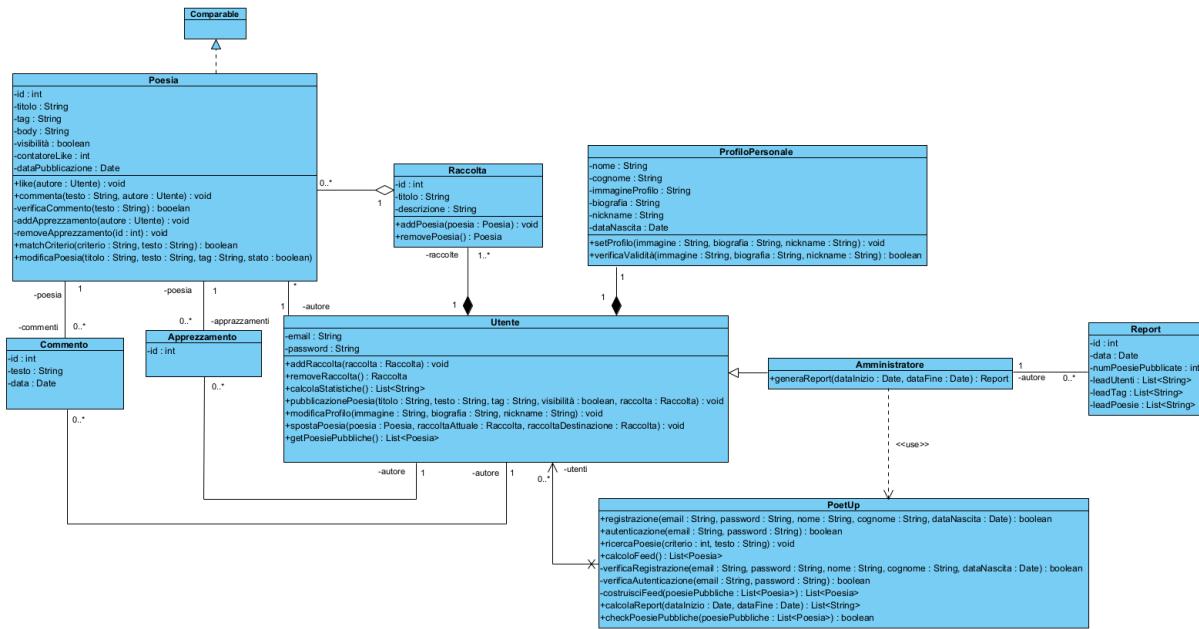


2.7.11 RicercaPoesie



Si è pensato di introdurre un metodo `getPoesiePubbliche()` per ottenere una lista di poesie pubbliche e un `matchCriterio(criterio, testo)` per ricercare nella lista in base al criterio (tag, testo della poesia o autore) e al testo della mia ricerca una poesia

2.8 Diagramma delle classi raffinato



OBJ

3. Piano di Test Funzionale

Si intende progettare i casi di test funzionale con la tecnica del [Category Partition Testing](#).

3.1 Registrazione

Registrazione		
Nickname	Email	Password
<ul style="list-style-type: none"> Stringa di caratteri alfanumerici di lunghezza ≤ 28 Stringa di caratteri di lunghezza > 28 [ERROR] Stringa che contiene caratteri speciali [ERROR] Stringa vuota [ERROR] 	<ul style="list-style-type: none"> Stringa di caratteri alfanumerici con il simbolo @ Stringa di caratteri già registrata [ERROR] Stringa di caratteri alfanumerici senza il simbolo @ [ERROR] Stringa vuota [ERROR] 	<ul style="list-style-type: none"> Stringa di caratteri alfanumerici di lunghezza > 8 e ≤ 32 Stringa di caratteri alfanumerici più almeno un carattere speciale Stringa di caratteri alfanumerici contenente almeno un numero Stringa di caratteri senza un numero

		<p>[ERROR]</p> <ul style="list-style-type: none"> Stringa di caratteri senza un carattere speciale <p>[ERROR]</p> <ul style="list-style-type: none"> Stringa di caratteri alfanumerici < 8 <p>[ERROR]</p> <ul style="list-style-type: none"> Stringa di caratteri alfanumerici > 32 <p>[ERROR]</p> <ul style="list-style-type: none"> Stringa vuota [ERROR]
--	--	--

Il numero di test da effettuarsi senza particolari vincoli è: $4 * 4 * 8 = 128$.

Con i vincoli **[ERROR]**, invece, il numero di test da eseguire per testare singolarmente i vincoli è 11 (3 per Nickname, 3 per Email e 5 per Password).

Il numero di test risultante è: $(1*1*1) + 11 = 12$.

TEST SUITE						
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese
1	Tutti input validi	Nickname valido, Email valida, Password valido		{Nickname="MarioRed02", Email: "MarioRossi02@gmail.com", Password: "Password1!"}	Utente registrato con successo	Il sistema mostra un messaggio per la registrazione avvenuta con successo e invia una notifica all'utente
2	Nickname stringa > 28 caratteri	Nickname stringa > 28 caratteri [ERROR] , Email, Password validi		{Nickname="MarioReddd dddddddddd dddd02", Email: "MarioRossi02@gmail.com", Password: "Password1!"}	Nickname troppo lungo!	

3	Nickname Stringa che contiene caratteri speciali	Nickname stringa con simboli [ERROR], Email, Password validi		{Nickname="M@rio-Red02", Email: "MarioRossi02@gmail.com", Password: "Password!!!"}	Formato Nickname errato!	
4	Nickname stringa vuota	Nickname stringa vuota [ERROR], Email, Password validi		{Nickname="", Email: "MarioRossi02@gmail.com", Password: "Password!!!"}	Errore Nickname vuoto!	
5	Email Stringa di caratteri già registrati	Nickname valido, Email Stringa di caratteri già registrati [ERROR], Password valida		{Nickname="MarioRed02", Email: "MarioGiàReg@gmail.com", Password: "Password!!!"}	Email già registrata!	
6	Email stringa di caratteri alfanumerici senza il simbolo @	Nickname valido, Email stringa di caratteri alfanumerici senza il simbolo @ [ERROR], Password valida		{Nickname="MarioRed02", Email: "MarioRossi02.gmail.com", Password: "Password!!!"}	Formato Email errato!	
7	Email stringa vuota	Nickname valido, Email stringa vuota [ERROR], Password valida		{Nickname="MarioRed02", Email: "", Password: "Password!!!"}	Tutti i campi sono obbligatori!	
8	Password stringa senza caratteri speciali	Nickname, Email validi, Password stringa senza caratteri speciali [ERROR]		{Nickname="MarioRed02", Email: "MarioRossi02@gmail.com", Password: "Password!!"}	Formato Password errato!	

9	Password stringa di caratteri senza un numero	Nickname, Email validi, Password stringa di caratteri senza un numero [ERROR]		{Nickname="MarioRed02", Email: "MarioRossi02@gmail.co m", Password: "Password!"}	Formato Password errato!	
10	Password Stringa di caratteri alfanumerici di lunghezza <8	Nickname, Email validi, Password stringa di caratteri alfanumerici di lunghezza <8 [ERROR]		{Nickname="MarioRed02", Email: "MarioRossi02@gmail.co m", Password: "Passw"}	Formato Password errato!	
11	Password Stringa di caratteri alfanumerici di lunghezza >32	Nickname, Email validi, Password stringa di caratteri alfanumerici di lunghezza > 32 [ERROR]		{Nickname="MarioRed02", Email: "MarioRossi02@gmail.co m", Password: "Passwordddddddddd dddddccccccccc ddddd1!"}	Formato Password errato!	
12	Password Stringa vuota	Nickname, Email validi, Password stringa vuota [ERROR]		{Nickname="MarioRed02", Email: "MarioRossi02@gmail.co m", Password: ""}	Tutti i campi sono obbligatori!	

3.2 Autenticazione

Autenticazione	
Email	Password
<ul style="list-style-type: none"> • Stringa di caratteri alfanumerici con il simbolo @ • Stringa di caratteri alfanumerici senza il simbolo @ [ERROR] • Stringa di caratteri non registrata [ERROR] • Stringa Vuota [ERROR] 	<ul style="list-style-type: none"> • La Stringa di caratteri corrisponde alla password salvata • Stringa di caratteri alfanumerici più almeno un simbolo speciale • Stringa di caratteri alfanumerici contenente almeno un numero • La Stringa di caratteri non corrisponde alla password salvata [ERROR] • Stringa di caratteri senza un numero

	<p>[ERROR]</p> <ul style="list-style-type: none"> • Stringa di caratteri senza un simbolo speciale [ERROR] • Stringa di caratteri alfanumerici < 8 [ERROR] • Stringa di caratteri alfanumerici > 32 [ERROR] • String vuota [ERROR]
--	---

Il numero di test da effettuarsi senza particolari vincoli è: $4 * 9 = 36$.

Con i vincoli **[ERROR]**, invece, il numero di test da eseguire per testare singolarmente i vincoli è 9 (3 per Email e 6 per Password). Il numero di test risultante è: $(1*1) + 9 = 10$.

TEST SUITE						
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese
1	Tutti input validi	Email valida Password valida	Utente Registrato	{Email: "MarioRossi02@gmail.com", Password: "Password1!"}	Accesso avvenuto con successo	Il sistema mostra un messaggio per l'autenticazione avvenuta con successo
2	Email stringa di caratteri alfanumerici senza il simbolo @ [ERROR] , Password valida	Email stringa di caratteri alfanumerici senza il simbolo @ [ERROR] , Password valida		{Email: "MarioRossi02@gmail.com", Password: "Password1!"}	Formato Email errato!	
3	Email Stringa di caratteri non registrata [ERROR] , Password valida	Email stringa di caratteri non registrata [ERROR] , Password valida		{Email: "Mario02NonReg@gmail.com", Password: "Password1!"}	Errore email non registrata!	
4	Email Stringa vuota [ERROR] , Password valida	Email Stringa vuota [ERROR] , Password valida		{Email: "", Password: "Password1!"}	Tutti i campi sono obbligatori!	

5	Password Stringa di caratteri non corrispondente alla password salvata [ERROR]	Email valida, Password stringa di caratteri non corrispondente a quella salvata [ERROR]		{Email: “MarioRossi02@gmail.com”, Password: “PasswSbagl2?”}	La password inserita non è corretta!	
6	Password stringa senza caratteri speciali [ERROR]	Email valida, Password stringa senza caratteri speciali [ERROR]		{Email: “MarioRossi02@gmail.com”, Password: “Password!”}	Formato Password errato!	
7	Password stringa di caratteri senza un numero [ERROR]	Email valida, Password stringa di caratteri senza un numero [ERROR]		{Email: “MarioRossi02@gmail.com”, Password: “Password@”}	Formato Password errato!	
8	Password Stringa di caratteri alfanumerici di lunghezza <8 [ERROR]	Email valida, Password stringa di caratteri alfanumerici di lunghezza <8 [ERROR]		{Email: “MarioRossi02@gmail.com”, Password: “Passw”}	Formato Password errato!	
9	Password Stringa di caratteri alfanumerici di lunghezza > 32 [ERROR]	Email valida, Password stringa di caratteri alfanumerici di lunghezza > 32 [ERROR]		{Email: “MarioRossi02@gmail.com”, Password: “Passwordddddddddddddd dddddddddd dddd!!”}	Formato Password errato!	
10	Password Stringa vuota [ERROR]	Email valida, Password Stringa vuota [ERROR]		{Email: “MarioRossi02@gmail.com”, Password: “”}	Tutti i campi sono obbligatori	

3.3 Gestione Profilo Personale

Gestione Profilo Personale

Nome	Cognome	DataNascita	Biografia
<ul style="list-style-type: none"> Stringa di caratteri di lunghezza ≤ 40 Stringa di caratteri di lunghezza > 40 [ERROR] Stringa che contiene caratteri speciali [ERROR] 	<ul style="list-style-type: none"> Stringa di caratteri di lunghezza ≤ 40 Stringa di caratteri di lunghezza > 40 [ERROR] Stringa che contiene caratteri speciali [ERROR] 	<ul style="list-style-type: none"> Data uguale o precedente a quella odierna, e con formato valido (yyyy-mm-dd) Data successiva a quella odierna [ERROR] Data con formato non valido [ERROR] 	<ul style="list-style-type: none"> Stringa di caratteri di lunghezza ≤ 255 Stringa di caratteri di lunghezza > 255 [ERROR]

Il numero di test da effettuarsi senza particolari vincoli è: $3 * 3 * 3 * 2 = 54$.

Con i vincoli **[ERROR]**, invece, il numero di test da eseguire per testare singolarmente i vincoli è 7 (2 per Nome, 2 per Cognome 2 per DataNascita e 1 per Biografia). Il numero di test risultante è: $(1*1*1*1*1) + 7 = 8$.

TEST SUITE						
Test Cas e ID	Descrizio ne	Classi di equivalenza coperte	Pre-cond izioni	Input	Output attesi	Post-cond izioni attese
1	Tutti input validi	Nome valido, Cognome valido, DataNascita valida Biografia valida	Utente registrato	{Nome="Mario", Cognome="Rossi", DataNascita="2003-21-11", Biografia="Giovane aspirante poeta"}	Profilo modificato	Il sistema mostra il messaggio di modifica avvenuta con successo
2	Nome Stringa di caratteri di lunghezza > 40 [ERROR] , Cognome, DataNascita, Biografia validi	Nome Stringa di caratteri di lunghezza > 40 [ERROR] ,		{Nome="Marioooooooooooooooo", Cognome="Rossi", DataNascita="2003-21-11", Biografia="Giovane aspirante poeta"}	Nome troppo lungo!	
3	Nome Stringa che contiene caratteri speciali [ERROR] ,	Nome Stringa che contiene simboli che non sono caratteri speciali [ERROR] ,		{Nome="M@rio", Cognome="Rossi", DataNascita="2003/21/11", Biografia="Giovane aspirante poeta"}	Formato nome errato!	

3.4 PubblicazionePoesia

PubblicazionePoesia			
Titolo	Testo	Tag	Raccolta
<ul style="list-style-type: none"> • Stringa di caratteri di lunghezza ≤ 25 • Stringa di caratteri di lunghezza > 25 [ERROR] • Stringa che contiene caratteri speciali [ERROR] • Stringa vuota [ERROR] 	<ul style="list-style-type: none"> • Stringa di caratteri di lunghezza ≤ 500 • Stringa che contiene segni di punteggiatura • Stringa di caratteri di lunghezza > 500 [ERROR] • Stringa che contiene caratteri speciali che non sono segni di punteggiatura [ERROR] • Stringa vuota [ERROR] 	<ul style="list-style-type: none"> • Stringa di caratteri di lunghezza ≤ 255 • Stringa che non contiene caratteri speciali al di fuori di # • Stringa di caratteri di lunghezza > 255 [ERROR] • Stringa che contiene caratteri speciali al di fuori di # [ERROR] • Stringa vuota [ERROR] 	<ul style="list-style-type: none"> • Stringa di caratteri di lunghezza ≤ 25 • Stringa Valida e raccolta esistente • Stringa Valida e raccolta inesistente <p>[EXTEND: CreazioneRaccoIta]</p> <ul style="list-style-type: none"> • Stringa di caratteri di lunghezza > 25 [ERROR] • Stringa che contiene caratteri speciali [ERROR] • Stringa vuota [ERROR]

Il numero di test da effettuarsi senza particolari vincoli è: $4 * 5 * 5 * 6 = 600$.

Con i vincoli **[ERROR]**, invece, il numero di test da eseguire per testare singolarmente i vincoli è 12 (3 per Titolo, 3 per Testo, 3 per Tag e 3 per Raccolta).

Ci saranno 2 test differenti che garantiscono la validità degli input (raccolta esistente o inesistente), quindi: Il numero di test risultante è: $(2 * 1 * 1 * 1) + 12 = 14$.

TEST SUITE						
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese
I	Tutti input validi e raccolta esistente	Titolo valido, Testo valido, Tag valido, Raccolta valida	Raccolta di destinazione esistente	{Titolo="Mattina", Testo="M'illumino d'immenso", Tag="#Gioia", Raccolta="Allegria"}	Poesia creata e aggiunta alla raccolta	Il sistema mostra il messaggio di creazione avvenuta con successo e

						aggiunge la poesia alla raccolta
2	Tutti input validi e raccolta inesistente	Titolo valido, Testo valido, Tag valido, Raccolta valida ma non esistente [EXTEND: CreazioneRaccolta]		{Titolo="Soldati", Testo="Si sta come d'autunno sugli alberi le foglie", Tag="#Tristezza", Raccolta="Preferiti"}	Raccolta creata e poesia aggiunta ad essa	Il sistema mostra il messaggio di creazione avvenuta con successo, crea la raccolta e vi aggiunge la poesia
3	Titolo Stringa di caratteri di lunghezza > 25	Titolo Stringa di caratteri di lunghezza > 25 [ERROR], Testo, Tag, Raccolta validi		{Titolo="Mattinaaaaaaaaaaaaaaaa", Testo="M'illuminod'immenso", Tag="#Gioia", Raccolta="Allegria"}	Titolo troppo lungo!	
4	Titolo Stringa che contiene caratteri speciali	Titolo Stringa che contiene caratteri speciali [ERROR], Testo, Tag, Raccolta validi		{Titolo="M@ttina", Testo="M'illuminod'immenso", Tag="#Gioia", Raccolta="Allegria"}	Formato Titolo errato!	
5	Titolo Stringa vuota	Titolo Stringa vuota [ERROR], Testo, Tag, Raccolta validi		{Titolo="", Testo="M'illuminod'immenso", Tag="#Gioia", Raccolta="Allegria"}	Il Titolo non può essere vuoto!	
6	Testo Stringa di caratteri di lunghezza > 500	Titolo valido, Testo Stringa di caratteri di lunghezza > 500 [ERROR], Tag, Raccolta validi		{Titolo="Mattina", Testo="M'illuminod'immensooooo(x501)", Tag="#Gioia", Raccolta="Allegria"}	Testo troppo lungo!	
7	Testo Stringa che contiene caratteri	Titolo valido, Testo Stringa che contiene caratteri		{Titolo="Mattina", Testo="#M'illuminod'immenso",	Formato Testo errato!	

	speciali che non sono segni di punteggiatura [ERROR], Tag, Raccolta validi	speciali che non sono segni di punteggiatura [ERROR], Tag, Raccolta validi		Tag="#Gioia", Raccolta="Allegria"} 		
8	Testo Stringa vuota	Titolo valido, Testo Stringa vuota [ERROR], Tag, Raccolta validi		{Titolo="Mattina", Testo="", Tag="#Gioia", Raccolta="Allegria"}	Il Testo non può essere vuoto	
9	Tag Stringa di caratteri di lunghezza > 255	Titolo valido, Testo Valido, Tag Stringa di caratteri di lunghezza > 255 [ERROR], Raccolta valida		{Titolo="Mattina", Testo="M'illumino d'immenso", Tag="#Gioiaaa...(x256)", Raccolta="Allegria"}	Tag troppo lungo!	
10	Tag Stringa che non contiene caratteri speciali al di fuori di #	Titolo valido, Testo valido, Tag Stringa che non contiene caratteri speciali al di fuori di # [ERROR], Raccolta valida		{Titolo="Mattina", Testo="M'illumino d'immenso", Tag="#Gioi@", Raccolta="Allegria"}	Formato Tag errato!	
11	Tag Stringa vuota	Titolo valido, Testo valido, Tag Stringa vuota [ERROR], Raccolta valida		{Titolo="Mattina", Testo="M'illumino d'immenso", Tag="", Raccolta="Allegria"}	Il Tag non può essere vuoto!	
12	Raccolta Stringa di caratteri di lunghezza > 25	Titolo, Testo, Tag validi, Raccolta Stringa di caratteri di lunghezza > 25 [ERROR]		{Titolo="Mattina", Testo="M'illumino d'immenso", Tag="#Gioia", Raccolta="Allegriaaaaaaaaaaaaaaaaaaaaaaa"aaaa"} 	Raccolta troppo lunga!	
13	Raccolta Stringa che contiene	Titolo, Testo, Tag validi, Raccolta Stringa che		{Titolo="Mattina", Testo="M'illumino d'immenso",	Formato Raccolta non valido!	

	caratteri speciali	contiene caratteri speciali [ERROR]		Tag="#Gioia", Raccolta="Allegri@"} 		
14	Raccolta Stringa vuota	Titolo, Testo, Tag validi, Raccolta Stringa vuota [ERROR]		{Titolo="Mattina", Testo="M'illumino d'immenso", Tag="#Gioia", Raccolta=""}	Raccolta non può essere vuoto	

3.5 Gestione Raccolte

Avendo lo scenario flussi di esecuzione indipendenti è stato suddiviso in sotto-funzionalità, ognuna delle quali avrà la propria Tabella di Validazione e la propria Test Suite.

3.5.1 Creazione Raccolta

Creazione Raccolta	
Titolo	Descrizione
<ul style="list-style-type: none"> Stringa di caratteri di lunghezza <= 25 Stringa di caratteri di lunghezza > 25 [ERROR] Stringa che contiene caratteri speciali [ERROR] Stringa vuota [ERROR] 	<ul style="list-style-type: none"> Stringa di caratteri di lunghezza <= 255 Stringa di caratteri di lunghezza > 255 [ERROR]

Il numero di test da effettuarsi senza particolari vincoli è: $4 * 2 = 8$.

Con i vincoli [ERROR], invece, il numero di test da eseguire per testare singolarmente i vincoli è 4 (3 per Titolo e 1 per Descrizione). Il numero di test risultante è: $(1*1) + 4 = 5$.

TEST SUITE						
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese
1	Tutti input validi	Titolo valido, Descrizione valida		{Titolo="Allegria", Descrizione="L'Allegria è una raccolta di poesie di Giuseppe Ungaretti"}	La Raccolta viene creata	
2	Titolo Stringa di caratteri > 25 [ERROR], Descrizione valida	Titolo Stringa di caratteri > 25 [ERROR], Descrizione valida		{Titolo="Allegriaaaaaaaaaaaaaaaaaaaaa", Descrizione="L'Allegria è una raccolta di poesie di Giuseppe Ungaretti"}	Titolo troppo lungo!	

3	Titolo Stringa che contiene caratteri speciali	Titolo Stringa che contiene simboli che non sono caratteri [ERROR], Descrizione valida		{Titolo="Allegri@", Descrizione="L'Allegria è una raccolta di poesie di Giuseppe Ungaretti"}	Formato Titolo errato!	
4	Titolo Stringa vuota	Titolo Stringa vuota [ERROR], Descrizione valida		{Titolo="", Descrizione="L'Allegria è una raccolta di poesie di Giuseppe Ungaretti"}	Il Titolo non può essere vuoto	
5	Descrizione Stringa di caratteri > 255	Titolo valido, Descrizione Stringa di caratteri > 255 [ERROR]		{Titolo="Allegria", Descrizione="L'Allegria è una raccolta di poesie di Giuseppe Ungarettiiii... (x256)"} }	Descrizione troppo lunga!	

3.5.2 SpostaPoesiaInRaccolta

SpostaPoesiaInRaccolta	
RaccoltaDestinazione	
<ul style="list-style-type: none"> • Stringa di caratteri di lunghezza <= 25 • Raccolta Esistente • Raccolta Inesistente [ERROR] • Stringa di caratteri di lunghezza > 25 [ERROR] • Stringa che contiene caratteri speciali [ERROR] • Stringa vuota [ERROR] 	

Il numero di test da effettuarsi senza particolari vincoli è: 6.

Con i vincoli [ERROR], invece, il numero di test da eseguire per testare singolarmente i vincoli è 4 per RaccoltaDestinazione. Il numero di test risultante è: 1 + 4 = 5.

TEST SUITE						
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese
I	Tutti input validi	RaccoltaDestinazione valida		{RaccoltaDestinazione="Allegria"}	Poesia spostata con Successo !	La poesia viene rimossa dalla raccolta di partenza e

					aggiunta alla raccolta di destinazione
2	RaccoltaDestinazione Inesistente	RaccoltaDestinazione valido ma Raccolta Inesistente [ERROR]		{RaccoltaDestinazione="RacNonEsistente"}	Raccolta inesistente!
3	RaccoltaDestinazione Stringa di caratteri di lunghezza > 25	RaccoltaDestinazione Stringa di caratteri di lunghezza > 25 [ERROR]		{RaccoltaDestinazione="Alle griaaaaaaaaaaaaaaaaaaaaaaaa aaaaaaaaa"}	Titolo troppo lungo!
4	RaccoltaDestinazione Stringa che contiene caratteri speciali	RaccoltaDestinazione Stringa che contiene caratteri speciali [ERROR]		{RaccoltaDestinazione="Alle gri@@"}	Formato Raccolta destinazione errato!
5	RaccoltaDestinazione Stringa vuota	RaccoltaDestinazione Stringa vuota [ERROR]		{RaccoltaDestinazione=""}	Raccolta di destinazione non inserita!

3.5.3 ModificaRaccolta

ModificaRaccolta	
Titolo	Descrizione
<ul style="list-style-type: none"> Stringa di caratteri di lunghezza ≤ 25 Stringa di caratteri di lunghezza > 25 [ERROR] Stringa che contiene caratteri speciali [ERROR] Stringa vuota [ERROR] 	<ul style="list-style-type: none"> Stringa di caratteri di lunghezza ≤ 255 Stringa di caratteri di lunghezza > 255 [ERROR]

Il numero di test da effettuarsi senza particolari vincoli è: $4 * 2 = 8$.

Con i vincoli [ERROR], invece, il numero di test da eseguire per testare singolarmente i vincoli è 4 (3 per Titolo e 1 per Descrizione).

Il numero di test risultante è: $(1*1) + 4 = 5$.

TEST SUITE						
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese
1	Tutti input validi	Titolo valido, Descrizione valida		{Titolo="Allegria", Descrizione="L'Allegria" è una raccolta di poesie di Giuseppe Ungaretti"}	La Raccolta viene modificata	
2	Titolo Stringa di caratteri > 25	Titolo Stringa di caratteri > 25 [ERROR], Descrizione valida		{Titolo="Allegriaaaaaaaaaaaaaaaaaaaaaaa", Descrizione="L'Allegria" è una raccolta di poesie di Giuseppe Ungaretti"}	Titolo troppo lungo!	
3	Titolo Stringa che contiene caratteri speciali [ERROR], Descrizione valida	Titolo Stringa che contiene caratteri speciali [ERROR], Descrizione valida		{Titolo="Allegri@", Descrizione="L'Allegria" è una raccolta di poesie di Giuseppe Ungaretti"}	Formato Titolo errato!	
4	Titolo Stringa vuota [ERROR], Descrizione valida	Titolo Stringa vuota [ERROR], Descrizione valida		{Titolo="", Descrizione="L'Allegria" è una raccolta di poesie di Giuseppe Ungaretti"}	Il Titolo non può essere vuoto	
5	Descrizione Stringa di caratteri > 255	Titolo valido, Descrizione Stringa di caratteri > 255 [ERROR]		{Titolo="Allegria", Descrizione="L'Allegria" è una raccolta di poesie di Giuseppe Ungarettiiiiiiiiii... (x256)"}	Descrizione troppo lunga!	

3.6 Commento Poesia

CommentoPoesia
Testo
<ul style="list-style-type: none">• Stringa di caratteri di lunghezza <= 255• Stringa di caratteri di lunghezza > 255 [ERROR]• Stringa vuota [ERROR]

Il numero di test da effettuarsi senza particolari vincoli è: 3.

Con i vincoli **[ERROR]**, invece, il numero di test da eseguire per testare singolarmente i vincoli è 2 per CommentoPoesia.

Il numero di test risultante è: 1 + 2 = 3.

TEST SUITE						
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese
1	Tutti input validi	Testo valido		{Testo="Mi sono emozionato leggendo questa poesia, semplicemente sensazionale, grazie per averla condivisa"}	Commento postato con successo!	Il commento viene inserito nella sezione commenti della poesia di riferimento
2	Testo Stringa di caratteri di lunghezza > 255	Testo Stringa di caratteri di lunghezza > 255 [ERROR]		{Testo="Mi sono emozionato leggendo questa poesia, semplicemente sensazionale, grazie per averla condivisaaaa...(x256)"}	Testo troppo lungo!	
3	Testo Stringa vuota	Testo Stringa vuota [ERROR]		{Testo=""}	Testo vuoto!	

3.7 GeneraReport

GeneraReport	
DataInizio	DataFine
<ul style="list-style-type: none"> • Data uguale o precedente a quella odierna, e con formato valido (yyyy-mm-dd) • Data successiva a quella odierna [ERROR] 	<ul style="list-style-type: none"> • Data uguale o precedente a quella odierna, e con formato valido (yyyy-mm-dd) • Data superiore a quella di Inizio • Data inferiore a quella di Inizio [ERROR]

<ul style="list-style-type: none"> Data con formato non valido [ERROR] 	<ul style="list-style-type: none"> Data successiva a quella odierna [ERROR] Data con formato non valido [ERROR]
---	---

Il numero di test da effettuarsi senza particolari vincoli è: $3 * 5 = 15$.

Con i vincoli [ERROR], invece, il numero di test da eseguire per testare singolarmente i vincoli è 5 (2 per Titolo e 3 per Descrizione).

Il numero di test risultante è: $(1*1) + 5 = 6$.

TEST SUITE						
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese
1	Tutti input validi	DataInizio valida, DataFine valida		{DataInizio="2010-05-10", DataFine="2010-05-17"}	Report generato con successo!	Il sistema genera i dati del report e li mostra sull'interfaccia
2	DataInizio successiva a quella odierna [ERROR], DataFine valida	DataInizio successiva a quella odierna [ERROR], DataFine valida		{DataInizio="2100-05-10", DataFine="2010-05-17"}	DataInizio errata!	
3	DataInizio con formato non valido [ERROR], DataFine valida	DataInizio con formato non valido [ERROR], DataFine valida		{DataInizio="10-05-2010", DataFine="2010-05-17"}	Formato DataInizio non valido!	
4	DataFine inferiore a DataInizio	DataInizio valida, DataFine inferiore a DataInizio [ERROR]		{DataInizio="2010-05-10", DataFine="2009-05-17"}	La DataFine non può essere inferiore alla DataInizio!	
5	DataFine successiva a quella odierna	DataInizio valida, DataFine successiva a quella odierna [ERROR]		{DataInizio="2010-05-10", DataFine="2100-05-17"}	DataFine errata!	

6	DataFine con formato non valido	DataInizio valida, DataFine con formato non valido [ERROR]		{DataInizio="2010-05-10", DataFine="17-05-2010"}	Formato DataFine non valido!	
---	---------------------------------	--	--	--	------------------------------	--

3.8 RicercaPoesia

Lo scenario è stato diviso in tre sotto-funzioni, una per ogni criterio di ricerca.

3.8.1 RicercaPoesiaPerTag

RicercaPoesiaPerTag
RicercaTag
<ul style="list-style-type: none"> Stringa di caratteri di lunghezza <= 255 Stringa che non contiene caratteri speciali al di fuori di # Stringa di caratteri di lunghezza > 255 [ERROR] Stringa che contiene caratteri speciali al di fuori di # [ERROR] Stringa vuota [ERROR]

Il numero di test da effettuarsi senza particolari vincoli è: 5.

Con i vincoli [ERROR], invece, il numero di test da eseguire per testare singolarmente i vincoli è 3 per RicercaTag.

Il numero di test risultante è: 1 + 3 = 4.

TEST SUITE						
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese
1	Tutti input validi	RicercaTag valida		{RicercaTag="Amore"}	Mostra Risultati ricerca!	
2	RicercaTag Stringa di caratteri di lunghezza > 255 [ERROR]	RicercaTag Stringa di caratteri di lunghezza > 255 [ERROR]		{RicercaTag="Amoreeeeeeeeeeee...((x256))"}	Ricerca troppo lunga!	
3	RicercaTag Stringa che contiene	RicercaTag Stringa che contiene		{RicercaTag="#Amore@@"}	Formato Ricerca errato!	

	caratteri che non sono segni di punteggiatura	caratteri speciali al di fuori di # [ERROR]			
4	RicercaTag Stringa vuota	RicercaTag Stringa vuota [ERROR]		{RicercaTag=""}	La Ricerca non può essere vuota!

3.8.2 RicercaPoesiaPerTesto

RicercaPoesiaPerTesto
RicercaTesto
<ul style="list-style-type: none"> Stringa di caratteri di lunghezza <= 500 Stringa che contiene segni di punteggiatura Stringa di caratteri di lunghezza > 500 [ERROR] Stringa che contiene caratteri speciali che non sono segni di punteggiatura [ERROR] Stringa vuota [ERROR]

Il numero di test da effettuarsi senza particolari vincoli è: 5.

Con i vincoli [ERROR], invece, il numero di test da eseguire per testare singolarmente i vincoli è 3 per Ricerca Testo.

Il numero di test risultante è: 1 + 3 = 4.

TEST SUITE						
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese
1	Tutti input validi	RicercaTesto valida		{RicercaTesto="M'illumin o D'immenso"}	Mostra Risultati ricerca!	
2	RicercaTesto Stringa di caratteri di lunghezza > 255	RicercaTesto Stringa di caratteri di lunghezza > 500 [ERROR]		{RicercaTesto="M'illumin o D'immensooooo... (x256)"}	Ricerca troppo lunga!	
3	RicercaTesto Stringa che contiene caratteri che non sono segni di	RicercaTesto Stringa che contiene caratteri che non sono segni di		{RicercaTesto="M'illumin o D'immenso@@"}	Formato Ricerca errato!	

	sono segni di punteggiatura	punteggjatura [ERROR]				
4	RicercaTesto Stringa vuota	RicercaTesto Stringa vuota [ERROR]		{RicercaTesto=""}	La Ricerca non può essere vuota!	

3.8.3 RicercaPoesiaPerTitolo

RicercaPoesiaPerTitolo
RicercaTitolo
<ul style="list-style-type: none"> Stringa di caratteri di lunghezza <= 25 Stringa di caratteri di lunghezza > 25 [ERROR] Stringa che contiene caratteri speciali [ERROR] Stringa vuota [ERROR]

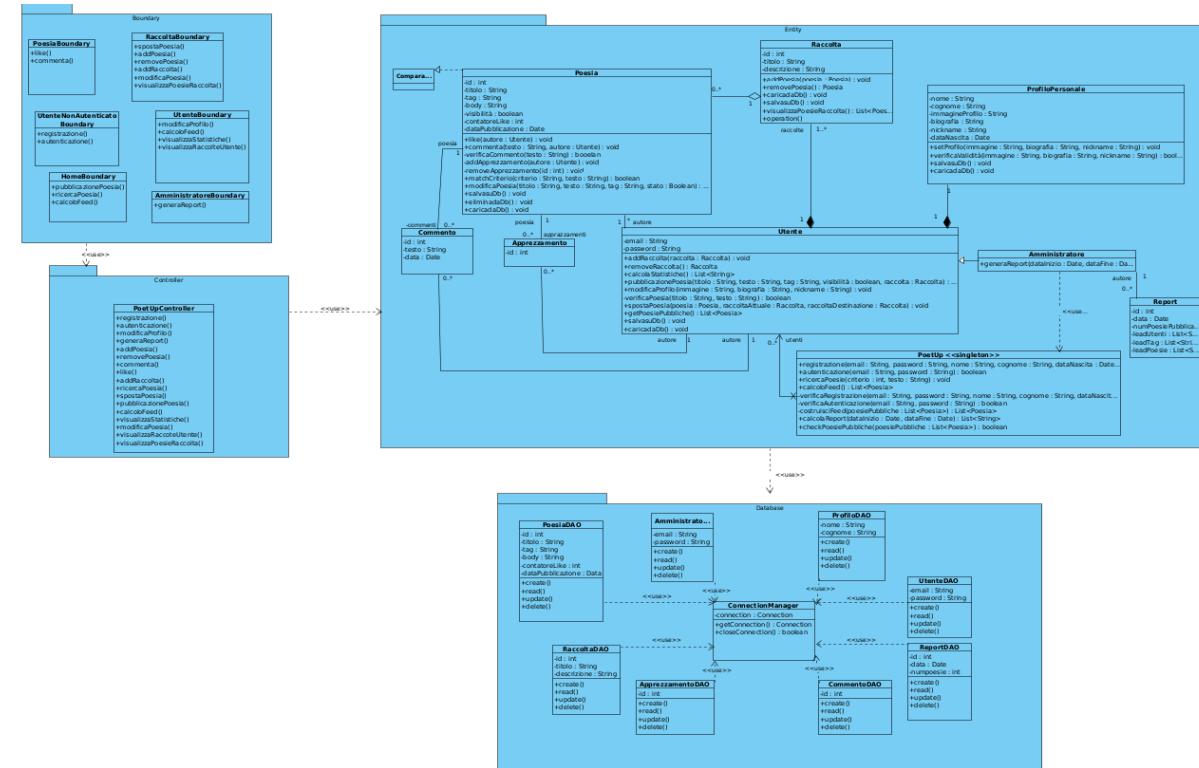
Il numero di test da effettuarsi senza particolari vincoli è: 4.

Con i vincoli [ERROR], invece, il numero di test da eseguire per testare singolarmente i vincoli è 3 per RicercaTitolo. Il numero di test risultante è: 1 + 3 = 4.

TEST SUITE						
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese
1	Tutti input validi	RicercaTitolo valida		{RicercaTitolo="Mattina"}	Mostra Risultati ricerca!	
2	RicercaTitolo Stringa di caratteri di lunghezza > 25 [ERROR]	RicercaTitolo Stringa di caratteri di lunghezza > 25 [ERROR]		{RicercaTitolo="Mattinaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"}	Ricerca troppo lunga!	
3	RicercaTitolo Stringa che contiene caratteri speciali [ERROR]	RicercaTitolo Stringa che contiene caratteri speciali [ERROR]		{RicercaTitolo="M@ttina"}	Formato Ricerca errato!	
4	RicercaTitolo Stringa vuota	RicercaTitolo Stringa vuota [ERROR]		{RicercaTitolo=""}	La Ricerca non può essere vuota!	

4. Progettazione

4.1 Class Diagram BCED



Il diagramma delle classi riportato sopra rappresenta una versione preliminare. A seguito di diverse scelte progettuali, la struttura è stata modificata e arricchita con nuovi metodi e package, introdotti per supportare al meglio il funzionamento del sistema.

4.1.1 Traduzione classi ed associazioni

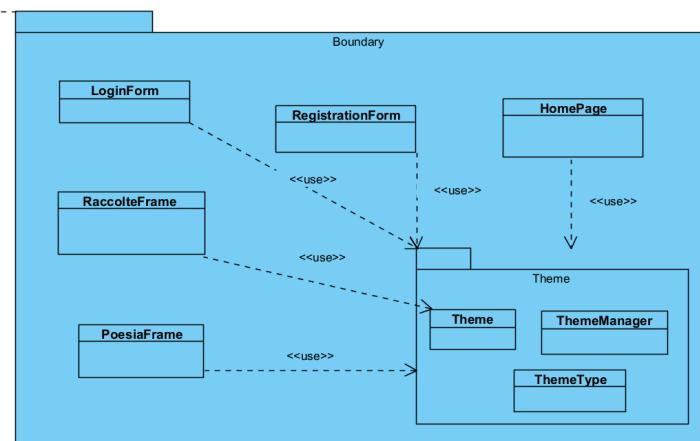
- Le classi `ProfiloPersonale` e `Raccolta` sono contenute nella classe `Utente`, in quanto esistono esclusivamente in funzione di essa. Entrambe conservano infatti un riferimento al relativo utente.
- Ogni utente possiede un singolo `ProfiloPersonale` e una collezione di oggetti `Raccolta`.
- All'utente è associato un singolo `ProfiloPersonale` e una molteplicità di raccolte.
- Le classi `Apprezzamento`, `Commento` e `Poesia` presentano un riferimento ad un singolo utente, loro `creator`, attraverso l'attributo `autore`.
- La classe `Amministratore` estende la classe `Utente` e, a differenza di quest'ultimo, può generare dei `Report`. Un report, a sua volta, conserva un riferimento ad un amministratore mediante un attributo `autore`.

- Una **poesia** è associata, attraverso un accoppiamento lasco, ad una **raccolta**. Ogni poesia possiede un riferimento alla raccolta della quale fa parte. Una raccolta mantiene dei riferimenti ad una lista di poesie che la compongono.
- La classe **Poesia** implementa l'interfaccia **comparable** istituendo così un ordinamento naturale in base alla **dataPubblicazione**, utile alla funzione **costruisciFeed()** per comparare le varie poesie in base all'attributo, così anche la classe **Commento**.
- Ogni poesia è associata a 0 o a N commenti e apprezzamenti. Tramite l'attributo **contatoreLike** una poesia ha traccia del numero di apprezzamenti ricevuti.

4.1.2 Pattern BCED

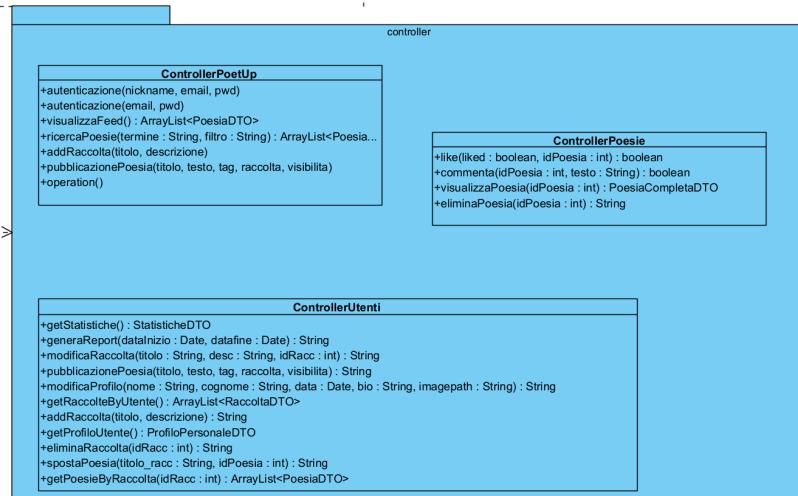
4.1.2.1 Package Boundary

Il package **Boundary** contiene tutti gli oggetti responsabili dell'interfaccia utente e della logica di presentazione; a questo livello tutte le classi corrispondono a delle interfacce e i relativi attributi non sono altro che gli elementi che le compongono, visualizzati a video. Le classi al suo interno sono, inoltre, responsabili della validazione degli input dell'utente e dei relativi messaggi di errore.



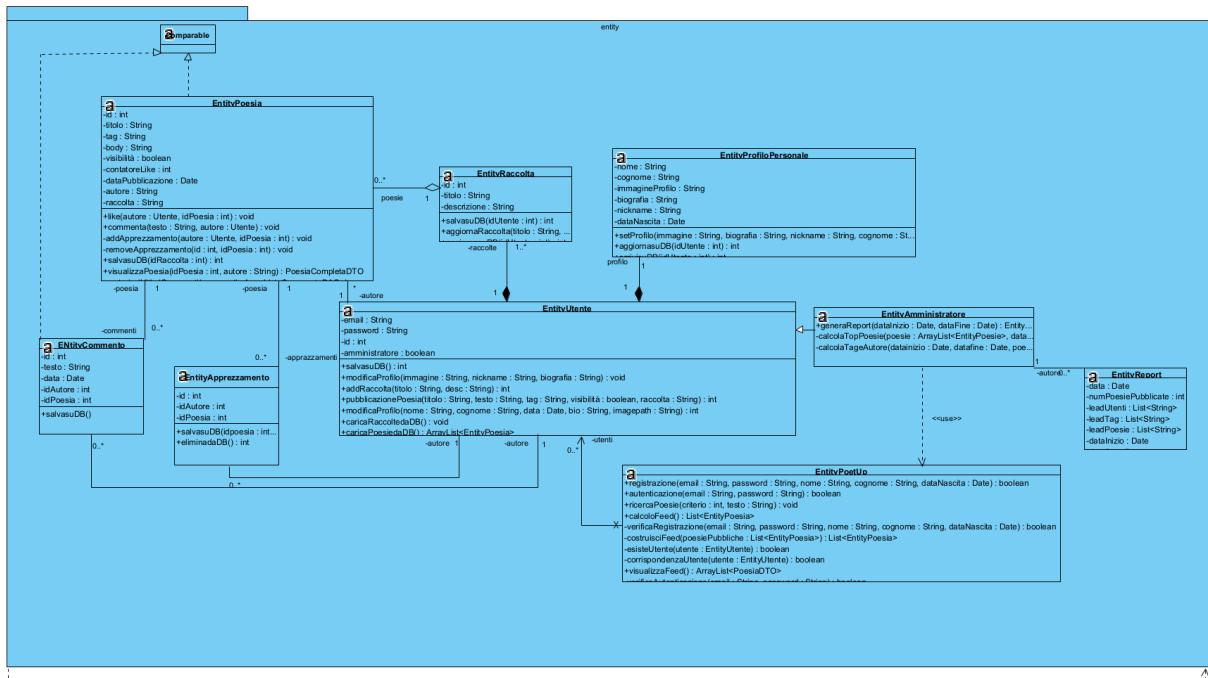
4.1.2.2 Package Controller

Questo package contiene le classi che percepiscono gli eventi generati dalle interazioni con l'interfaccia utente e ne delegano la gestione al package Entity.



4.1.2.3 Package Entity

Il Package **Entity** contiene tutti gli oggetti che rappresentano la semantica delle entità del dominio applicativo e corrispondono alle strutture dati presenti all'interno del database di persistenza.



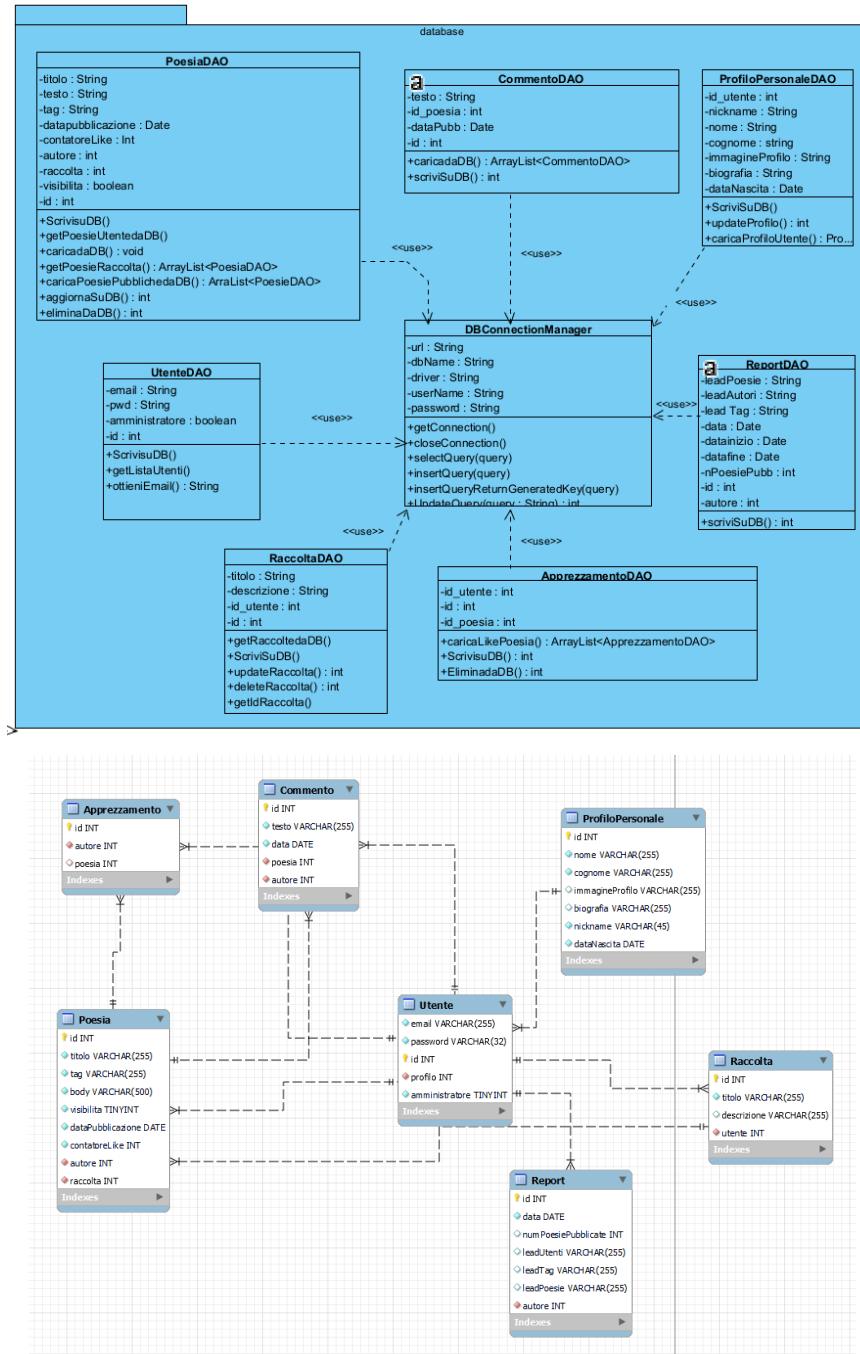
4.1.2.4 Package Database

Il Package **Database** contiene tutte le classi responsabili dell'estrazione dei dati dal DB, esponendo una vera e propria interfaccia che di fatto rende indipendenti le classi della Business Logic (Entity) dalla tecnologia di persistenza utilizzata.

In particolare, tra le strategie di risoluzione del problema dell'**impediment mismatch**, che nasce dalla mancata corrispondenza tra il modello Object Oriented e quello relazionale, si è deciso di

adottare quella delle classi **DAO** (Data Access Objects), che consiste nell'utilizzo di appositi oggetti per l'accesso ai dati.

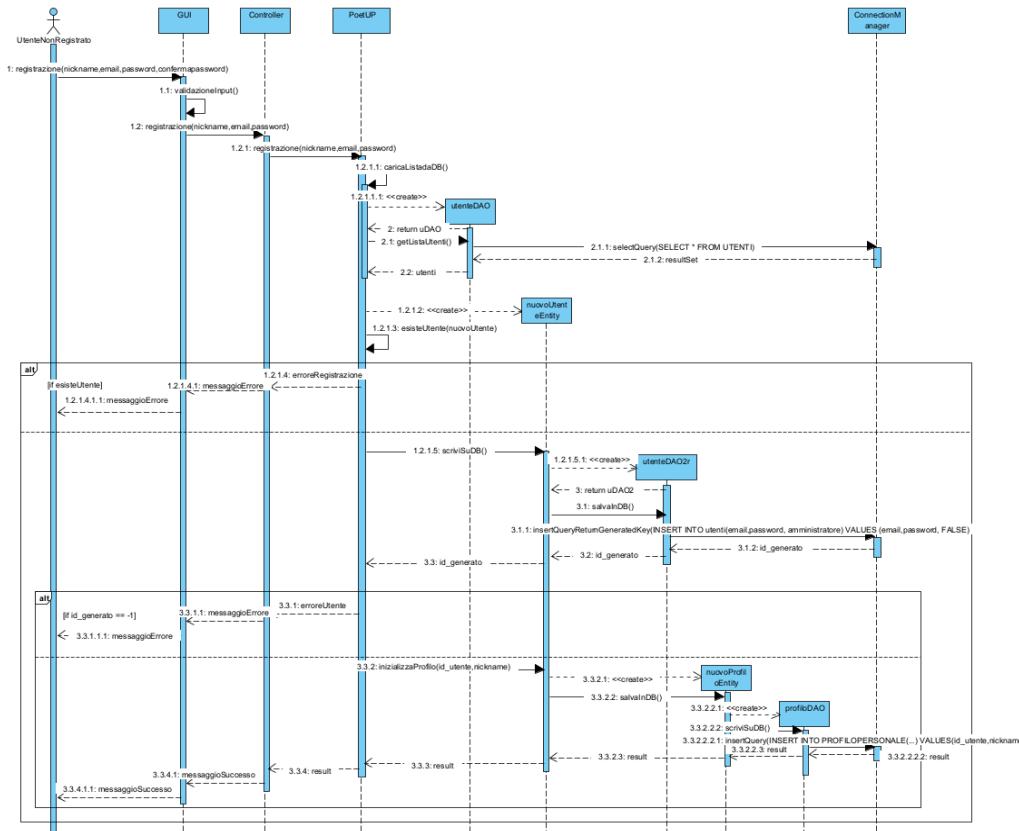
Ognuna di queste classi conterrà i metodi **CRUD** per l'interrogazione e la manipolazione della corrispondente classe di dominio (*query*), implementati in funzione di un'ulteriore classe, **DBConnectionManager**, che costituisce di fatto l'unico punto di accesso vero e proprio al DB, sfruttando i metodi che questa mette a disposizione.



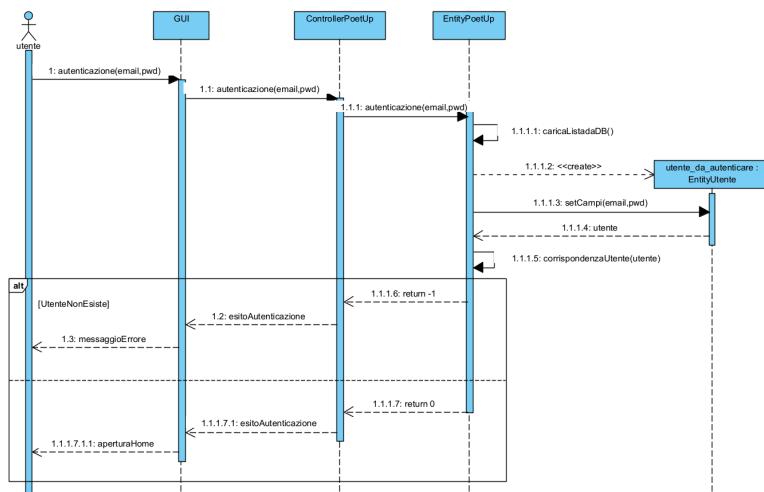
4.2 Sequence Diagram BCED

Per una visualizzazione più chiara e completa dei diagrammi di sequenza BCED, è consigliato aprirli utilizzando il progetto VPP fornito in allegato al documento.

4.2.1 Registrazione

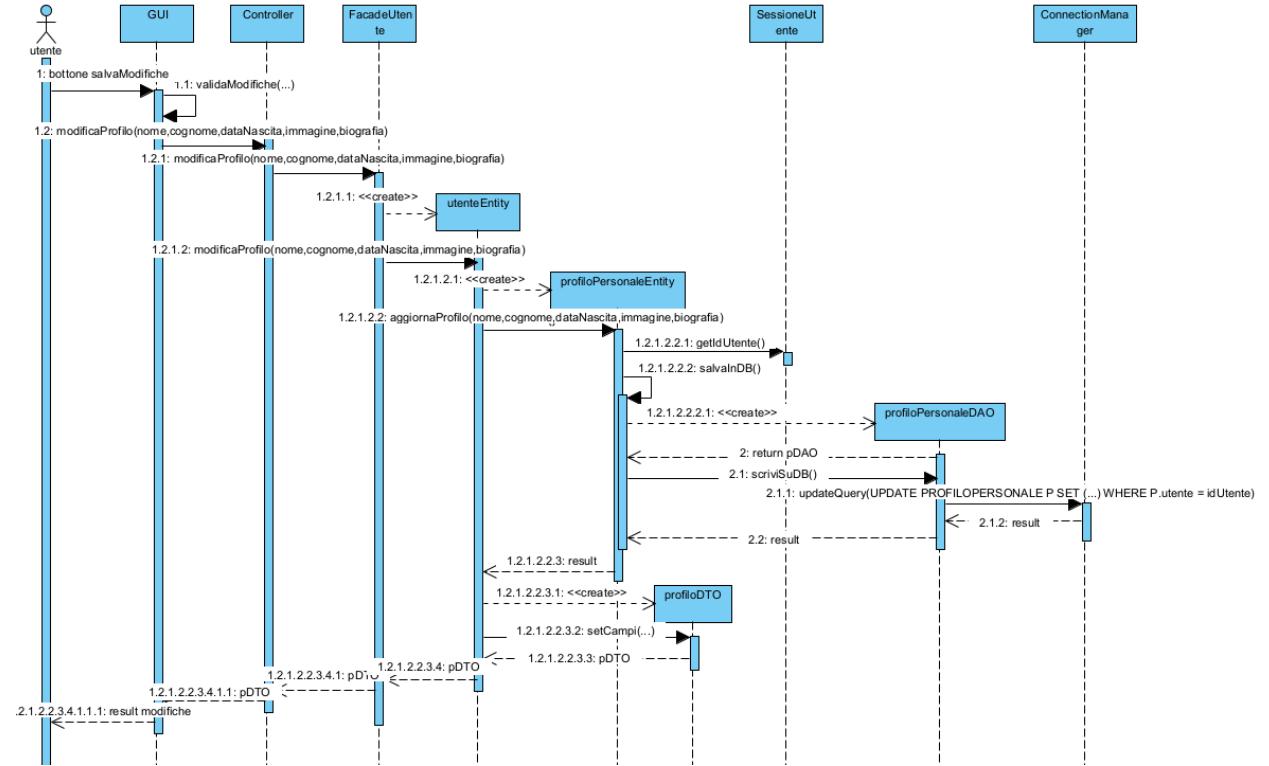


4.2.2 Autenticazione

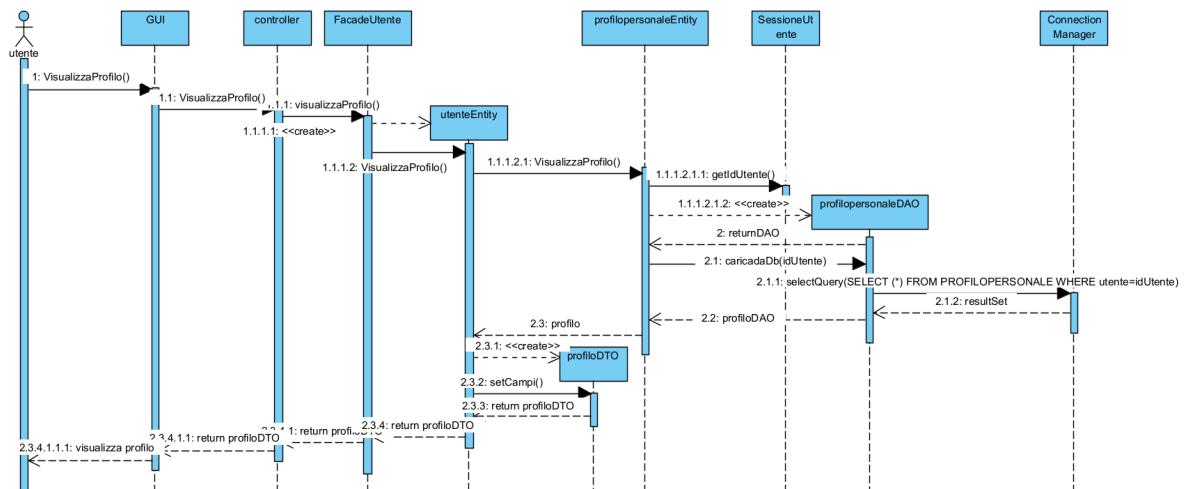


4.2.3 GestioneProfiloPersonale

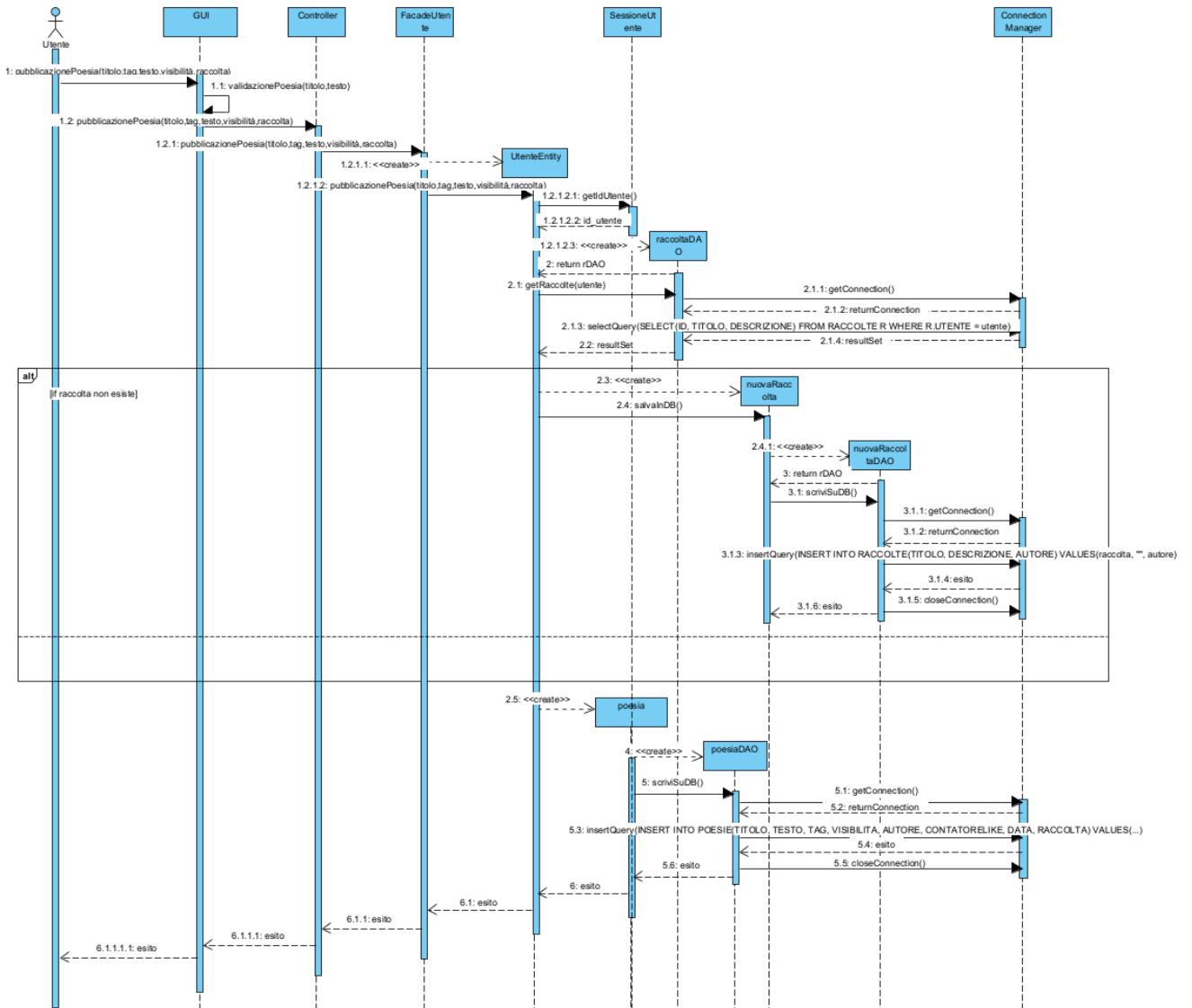
4.2.3.1 ModificaProfilo



4.2.3.2 VisualizzaProfilo

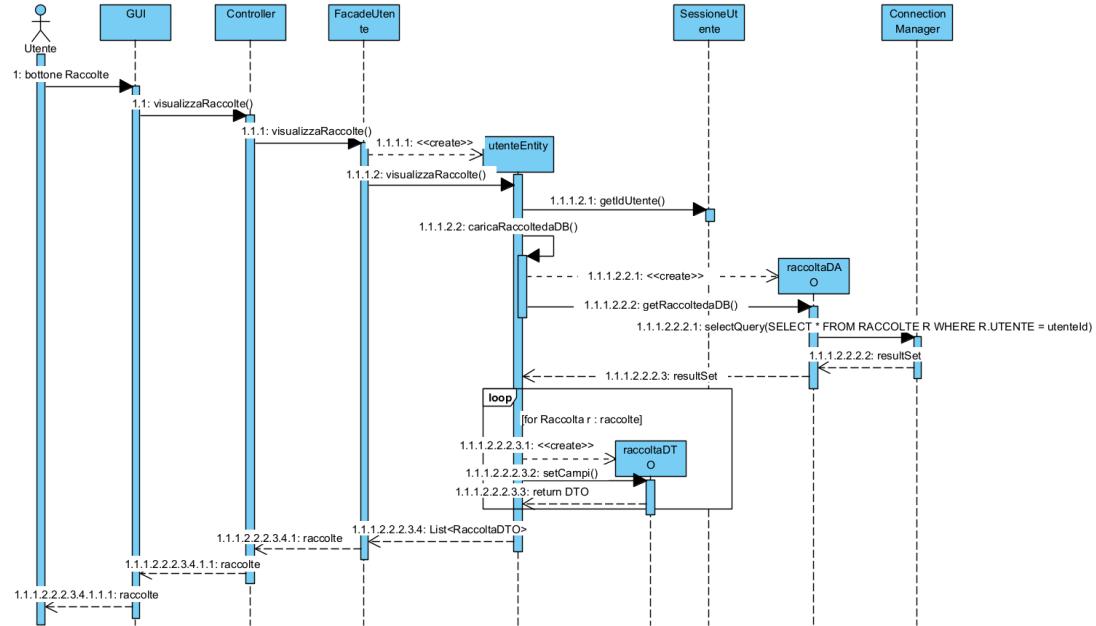


4.2.4 PubblicazionePoesia

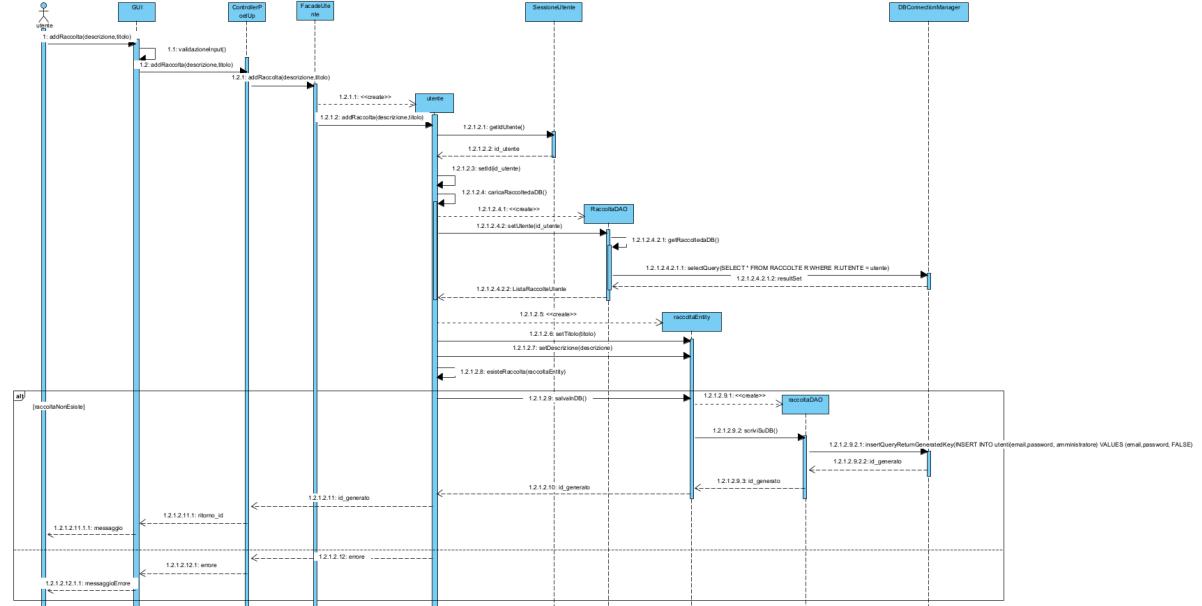


4.2.5 GestioneRaccolte

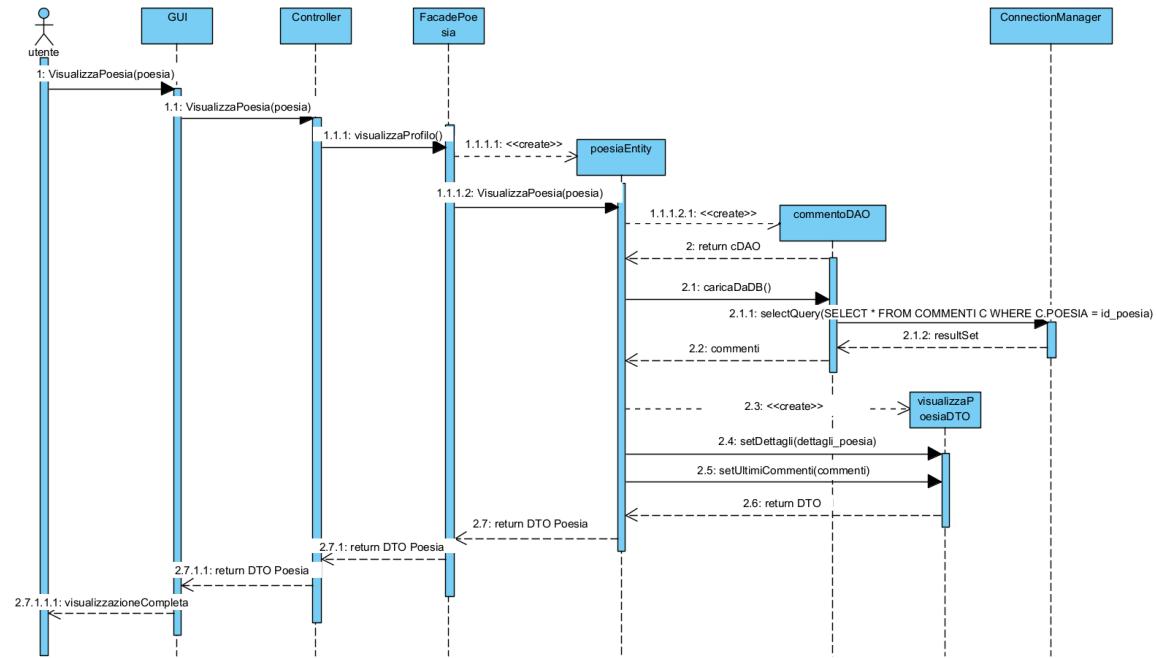
4.2.5.1 VisualizzaRaccolta



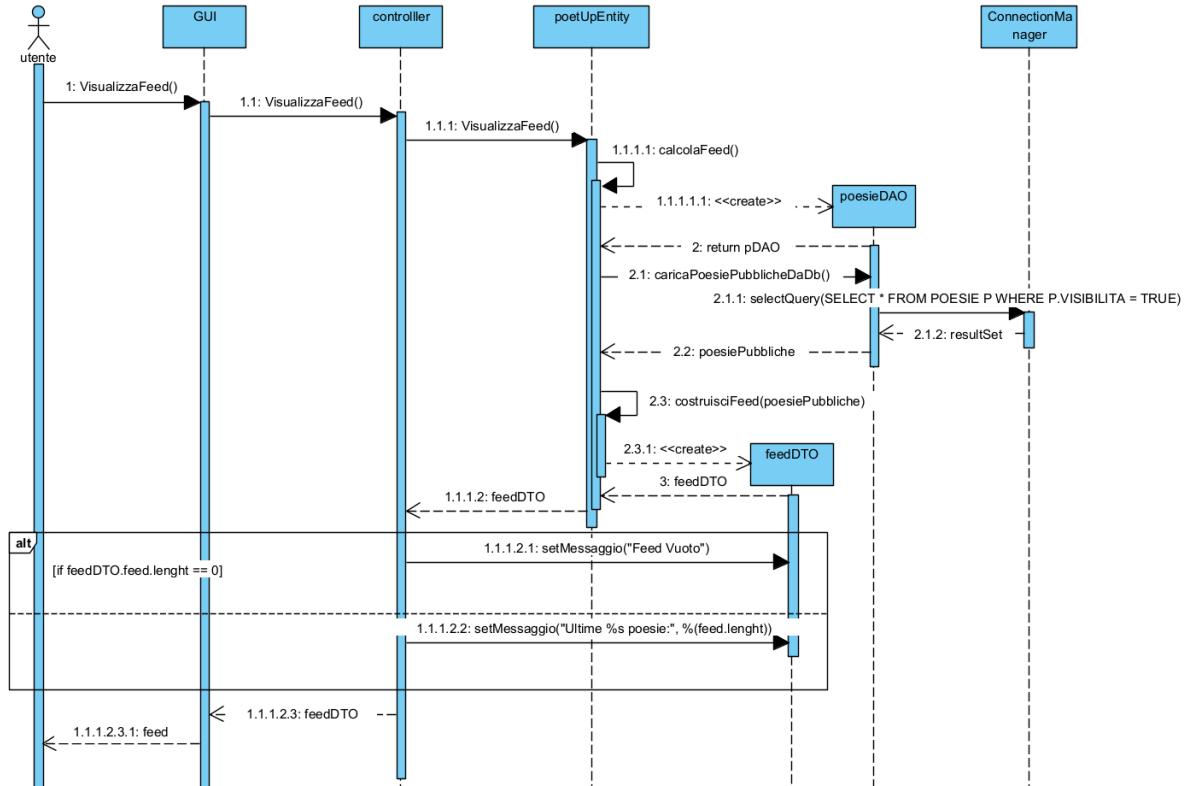
4.2.5.2 CreaRaccolta



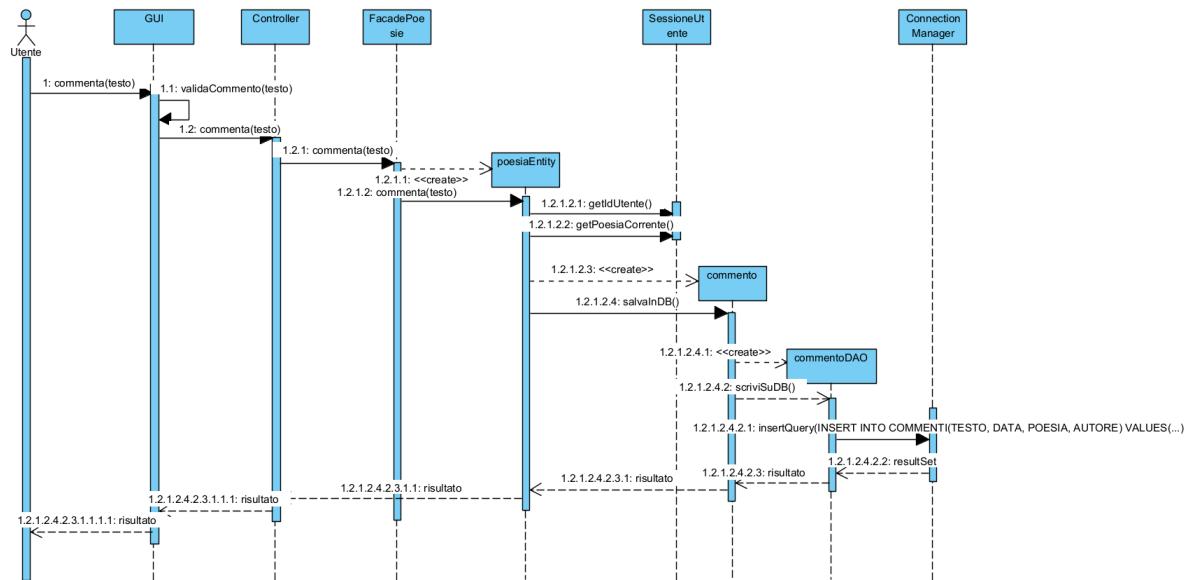
4.2.6 VisualizzazionePoesia



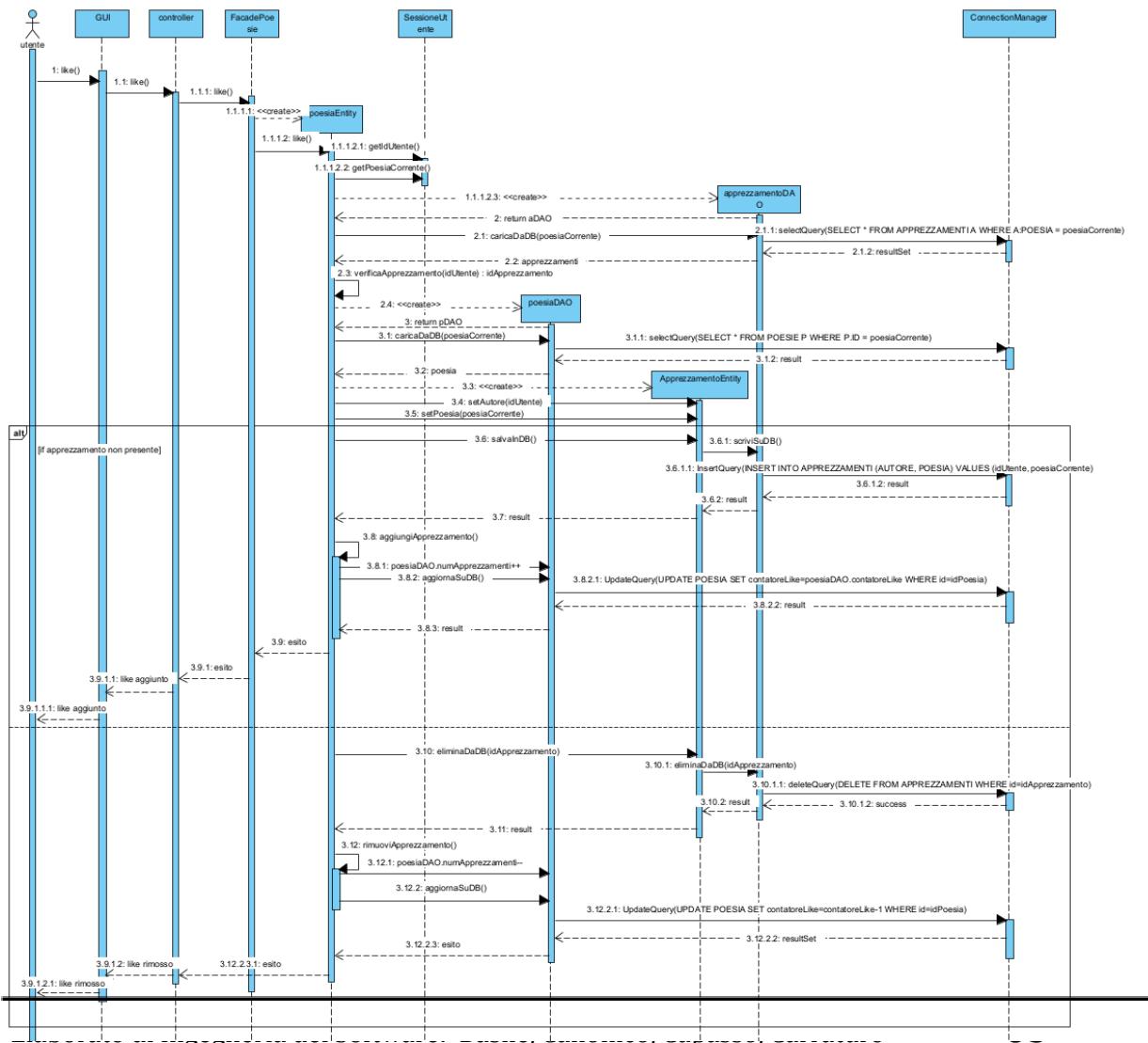
4.2.7 VisualizzazioneFeed



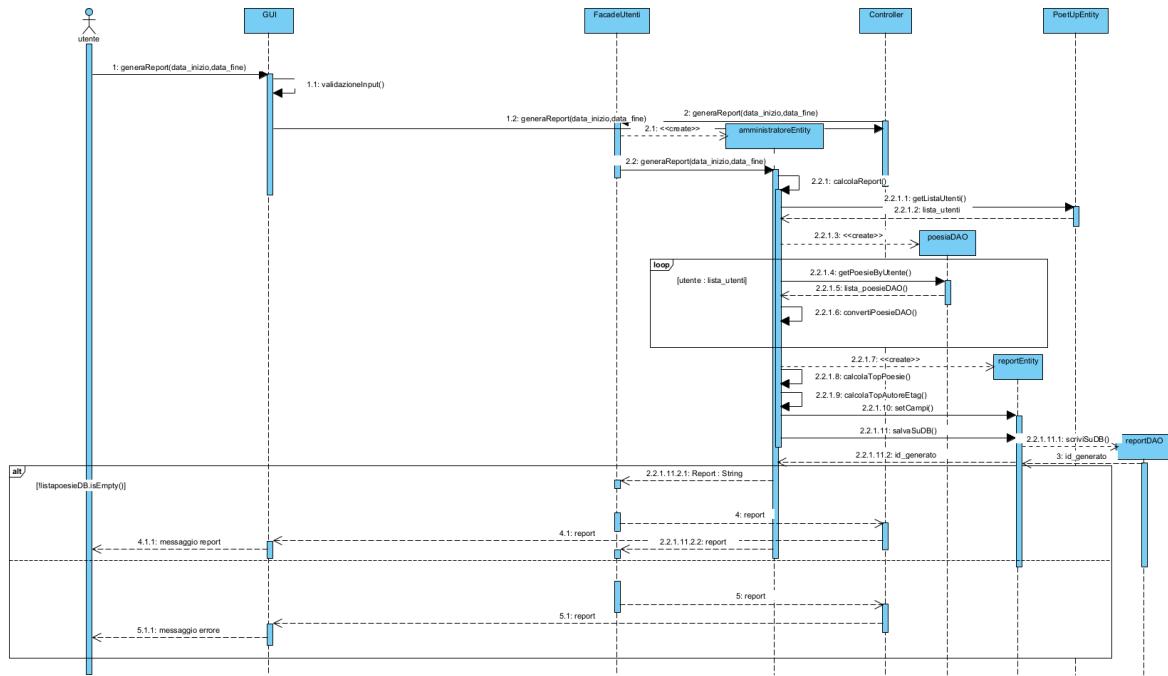
4.2.8 CommentoPoesia



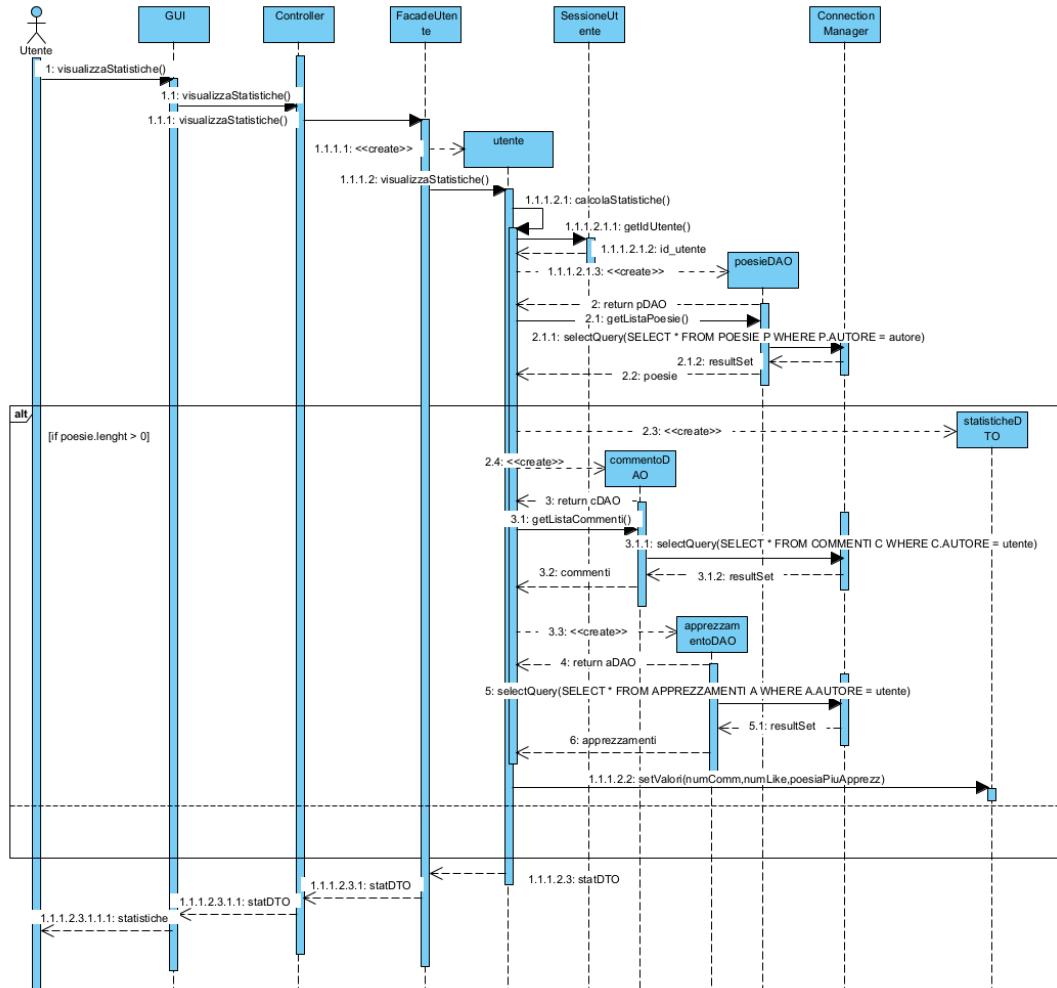
4.2.9 ApprezzamentoPoesia



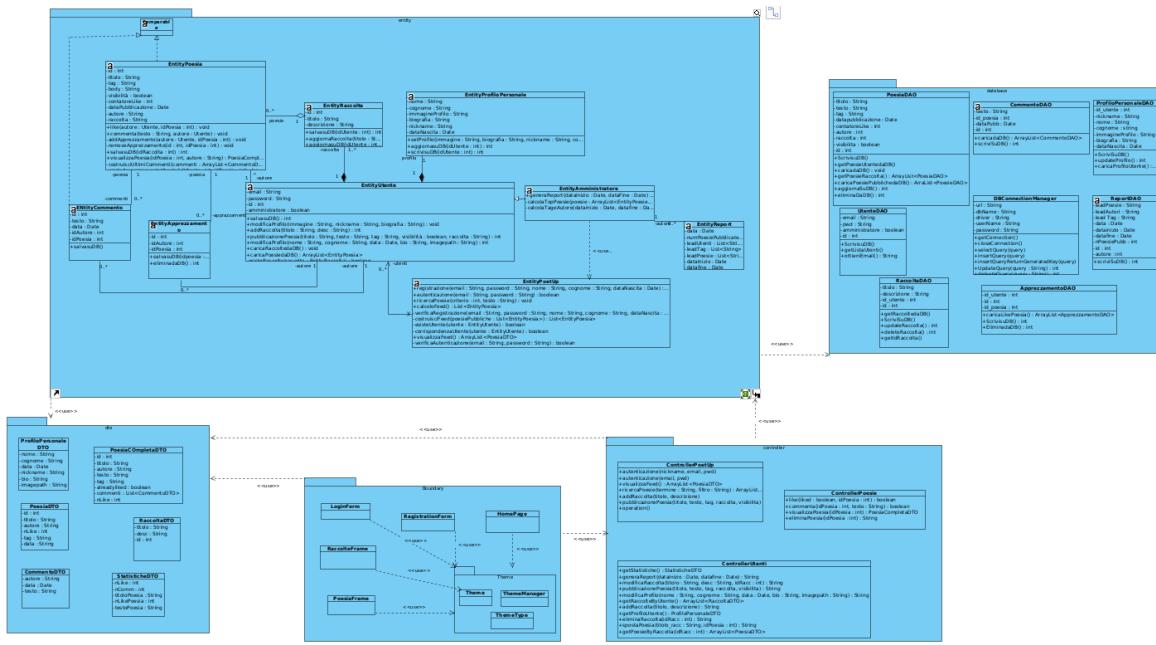
4.2.10 GeneraReport



4.2.11 VisualizzazioneStatistiche



4.3 Class Diagram Raffinato

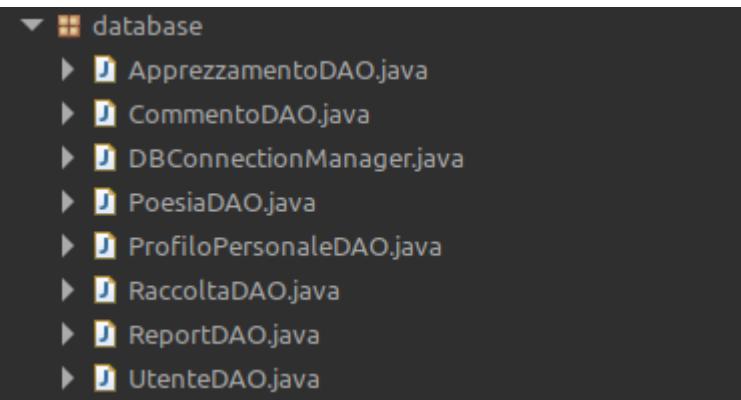


Per una visualizzazione più chiara e completa del class diagram , è consigliato visualizzarlo utilizzando il progetto VPP fornito in allegato al documento.

5. Implementazione

A partire dall'adozione del pattern BCED, si è ritenuto opportuno introdurre alcuni package di supporto, come [dto](#) e [session](#), per garantire il corretto funzionamento del software. Inoltre, sono stati aggiunti il package [email](#) e il sottopackage [theme](#) all'interno del package [boundary](#).

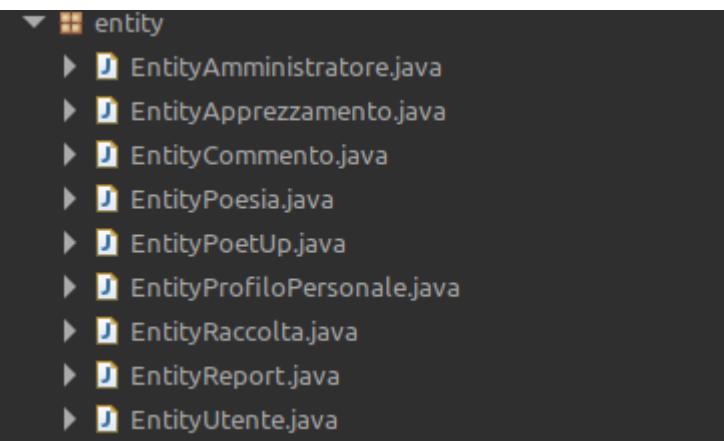
5.1 Package Database



Il package [database](#) rappresenta il livello di accesso e gestione persistente dei dati dell'applicazione. Contiene le classi responsabili dell'interazione diretta con il database, incaricate di eseguire operazioni **CRUD** (Create, Read, Update, Delete) sui dati relativi alle entità del dominio.

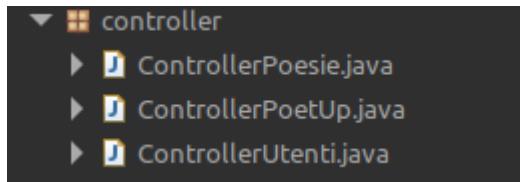
Le classi **DAO** encapsulano la logica di accesso ai dati per ciascuna entità, garantendo un'interfaccia disaccoppiata rispetto ai livelli superiori dell'applicazione. Il package include inoltre una classe, [DBConnectionManager](#), che centralizza le responsabilità di connessione al database.

5.2 Package Entity



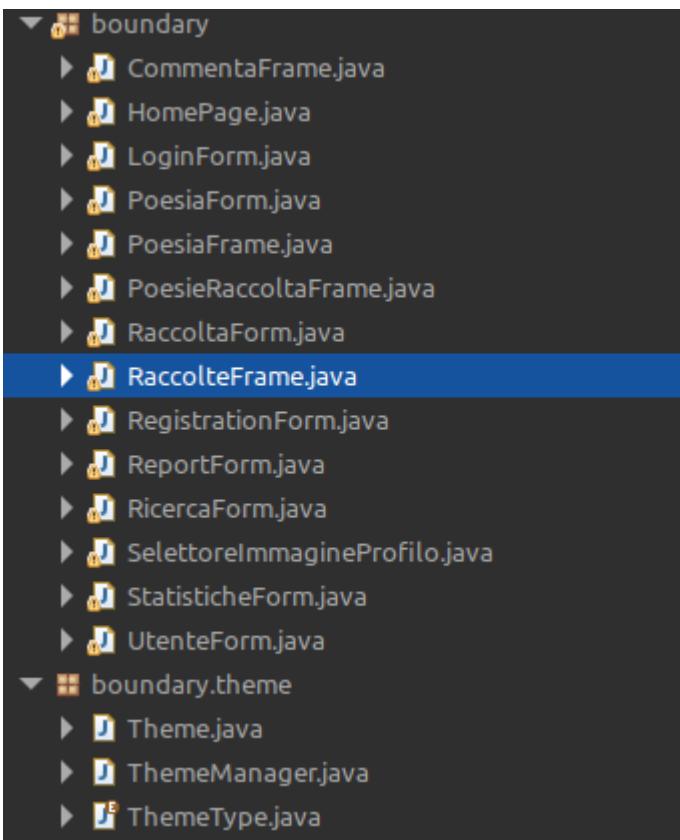
Il package [entity](#) raccoglie le classi che rappresentano i concetti fondamentali del dominio dell'applicazione, modellando gli oggetti persistenti e i loro attributi.

5.3 Package Controller



Il package [Controller](#) è stato suddiviso in tre classi per delegare in modo più chiaro le responsabilità per ciascuna entità, migliorando così la leggibilità e la manutenibilità del codice.

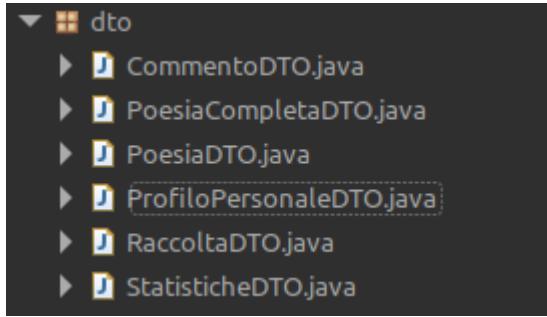
5.4 Package Boundary



L'interfaccia è stata strutturata in maniera modulare, ogni finestra si occupa di una o al più due responsabilità; questo ha migliorato l'estensibilità dell'interfaccia e la manutenibilità.

Inoltre è stato introdotto un sottopackage [theme](#) contenente le classi che si occupano della gestione del tema dell'applicazione, gestito come due array di [Color](#), utilizzate per definire i temi chiaro e scuro dell'applicazione, la classe [ThemeManager](#) è stata implementata seguendo il [pattern Singleton](#), garantendo l'unicità dell'istanza che gestisce la Palette da usare, evitando incongruenze tra le varie [JFrame](#).

5.4 Package DTO



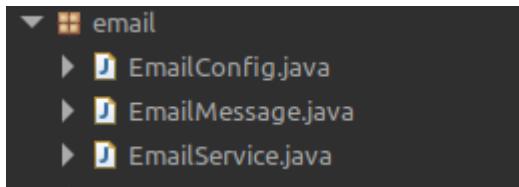
Package necessario per la visualizzazione tramite interfaccia degli attributi delle classi Entity. La corrispondenza degli attributi tra DTO ed Entity non è 1 : 1, dato che non tutti gli attributi dell'entity sono necessari nella visualizzazione. Inoltre, ogni classe DTO è stata dotata, per progettazione, di un costruttore vuoto e di un costruttore che inizializza tutti i suoi attributi.

5.5 Package Session



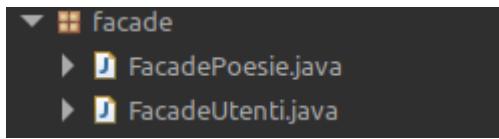
All'interno di questo package è presente una classe necessaria per mantenere alcune informazioni dell'utente che ha effettuato l'accesso e dei suoi privilegi. È principalmente utilizzata dal package Entity e dal package Boundary per ottenere un'interfaccia personalizzata in base ai permessi.

5.6 Package Email



Per il servizio di mailing, sono state utilizzate le classi presenti nel [package email](#).

5.7 Package Facade



Utilizzato per ridurre l'accoppiamento tra i package controller ed entity. Sono stati utilizzati un [FacadePoesie](#) e un [FacadeUtenti](#) per suddividere strutturalmente le responsabilità dell'utente da quelle della poesia.

6. Testing

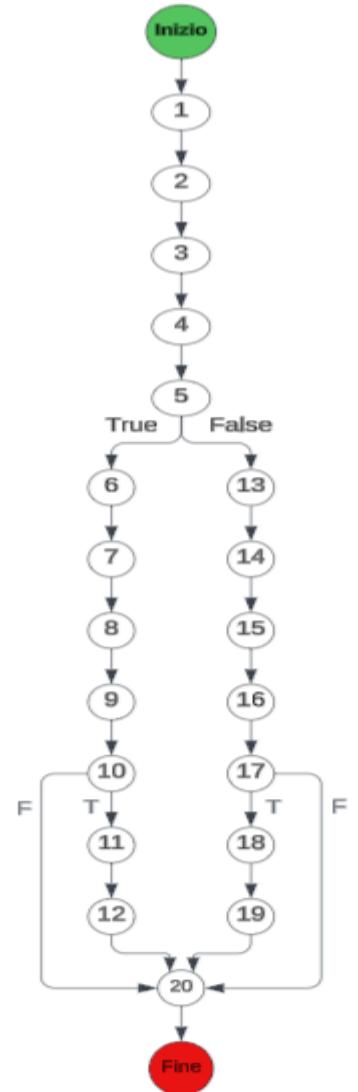
6.1 Test Strutturale

6.1.1 Complessità ciclomatica

Si intende costruire il Control Flow Graph per due dei metodi delle classi implementate e mostrare il calcolo del numero ciclomatico e i percorsi linearmente indipendenti.

6.1.1.1 like() - EntityPoesia

```
public boolean like(boolean liked, int idPoesia) {  
    boolean like_stato = false; (1)  
    int idUtente = SessioneUtente.getIdUtente(); (2)  
    this.setId(idPoesia); (3)  
    EntityApprezzamento Like = new EntityApprezzamento(); (4)  
    if (liked==false) { (5)  
        //aggiunge al DB  
        System.out.println(liked); (6)  
        Like.setId_autore(idUtente); (7)  
        Like.setId_poesia(idPoesia); (8)  
        int res_query1 = Like.salvaSuDB(this.getId()); (9)  
        if (res_query1 != -1) { (10)  
            this.aggiungiApprezzamento(Like, this.getId()); (11)  
            like_stato = true; (12)  
        }  
    } else {  
        //rimuove da DB  
        System.out.println(liked); (13)  
        Like.setId_autore(idUtente); (14)  
        Like.setId_poesia(idPoesia); (15)  
        int res_query2 = Like.eliminaDaDB(); (16)  
        if (res_query2 != -1) { (17)  
            this.rimuoviApprezzamento(Like, this.getId()); (18)  
            like_stato = false; (19)  
        }  
    }  
    return like_stato; (20) }
```



Il metodo studiato richiama a sua volta i metodi `salvaSuDB()`, `aggiungiApprezzamento()`, `eliminaDaDB()` e `rimuoviApprezzamento()` del livello Entity. Si noti però che `aggiungiApprezzamento` e `rimuoviApprezzamento` sono metodi sequenziali, quindi non contribuiscono alla complessità ciclomatica. Inoltre, `salvaSuDB()` e `eliminaDaDB` richiamano i metodi `scriviSuDB()` e `eliminaDaDB()` del livello database dei quali dovremo valutare la complessità ciclomatica.

Numero Ciclomatico, senza i metodi `scriviSuDB()` ed `eliminaDaDB()`:

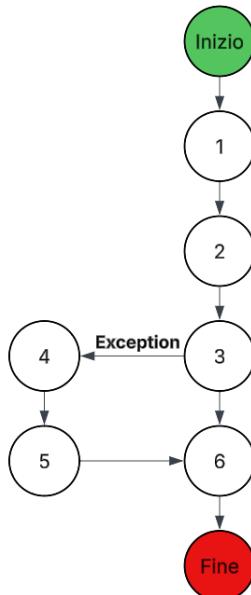
- Numero di regioni chiuse del grafo + 1 = 4
- Numero di nodi predicati (5, 10, 17) + 1 = 4
- #archi - #nodi + 2 = (22 - 20) + 2 = 4

Cammini:

- 1) 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 11 → 12 → 20
- 2) 1 → 2 → 3 → 4 → 5 → 6 → 7 → 8 → 9 → 10 → 20
- 3) 1 → 2 → 3 → 4 → 5 → 13 → 14 → 15 → 16 → 17 → 18 → 19 → 20
- 4) 1 → 2 → 3 → 4 → 5 → 13 → 14 → 15 → 16 → 17 → 20

6.1.1.1.1 ScriviSuDB() - ApprezzamentoDAO

```
public int ScriviSuDB() {
    int ret = 0; (1)
    String query = "INSERT INTO Apprezzamenti(autore, poesia) VALUES (" +
this.id_utente + ", " + this.id_poesia + ")"; (2)
    try {
        ret = DBConnectionManager.insertQueryReturnGeneratedKey(query); (3)
    } catch (ClassNotFoundException | SQLException e) {
        e.printStackTrace(); (4)
        ret = -1; (5)
    }
    return ret; (6)
}
```



Numero Ciclomatico:

- Numero di regioni chiuse del grafo + 1 = 2
- Numero di nodi predicati (1) + 1 = 2
- #archi - #nodi + 2 = (6 - 6) + 2 = 2

Cammini:

- 1) 1 → 2 → 3 → 6
- 2) 1 → 2 → 3 → 4 → 5 → 6

Numero Ciclomatico complessivo: $4 + 2 + 2 - 1 - 1 = 6$

I due -1 servono per evitare di contare due volte la complessità ciclomatica dei metodi `scriviSuDB()` ed `eliminaDaDB()`.

Nel computo totale, si è deciso di considerare la loro complessità come parte del metodo like(), quindi non si sommano nuovamente i + l che verrebbero normalmente aggiunti nel conteggio separato dei metodi. Per compensare questo, si sottrae l per ciascuno di quei metodi nel totale.

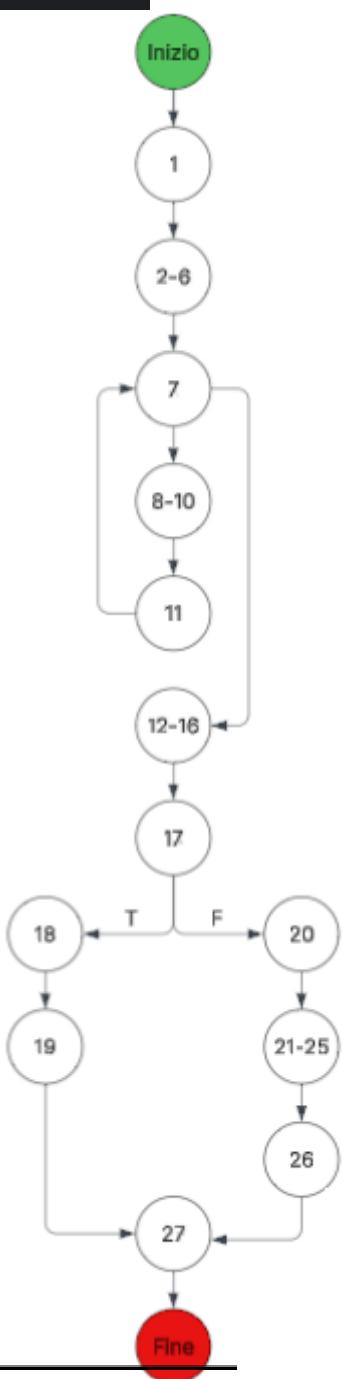
D'ora in avanti, considerata l'estrema semplicità dei CFG relativi ai metodi che interagiscono con il livello database, si darà per assunta la loro struttura e non verrà fornita un'analisi dettagliata di quelli composti esclusivamente da un blocco `try-catch`.

6.1.1.2 generaReport() - EntityAmministratore

```

public String generaReport(Date dataInizio, Date dataFine) {
    EntityReport report=new EntityReport(); (1)
    EntityPoetUp.setElencoUtenti(new ArrayList<EntityUtente>()); (2)
    EntityPoetUp.caricaListaDaDB(); (3)
    ArrayList<EntityUtente> utenti = EntityPoetUp.getElencoUtenti();
    ArrayList<EntityPoesia> poesieDB = new ArrayList<>(); (5)
    PoesiaDAO poesiaDAO = new PoesiaDAO(); (6)
        // Recupero poesie da DB per ogni utente
        for (EntityUtente utente : utenti) { (7)
            poesiaDAO.setAutore(utente.getId()); (8)
            ArrayList<PoesiaDAO> tutteLePoesie =
    poesiaDAO.getPoesieUtenteDaDB(); (9) (+2 decision points)
            ArrayList<EntityPoesia> tutteLePoesieEntity =
    convertiPoesieDAO(tutteLePoesie); (10)
            poesieDB.addAll(tutteLePoesieEntity); (11)
        }
        ArrayList<String>
    autore_tag=calcoloTageAutore(poesieDB,dataInizio,dataFine); (12)
        report.setLeadAutori(autore_tag.get(0)); (13)
        report.setLeadTag(autore_tag.get(1)); (14)
        // Trova poesie con più like e calcolo contatorePoesie
        calcolaTopPoesie(poesieDB,dataInizio,dataFine,report); (15)
        StringBuilder report_string= new StringBuilder(); (16)
        if(report.salvasuDB()==-1) { (17) (+1 decision point)
            report_string.append(">>> REPORT <<<\n"); (18)
            report_string.append("ERRORE NEL SALVATAGGIO DEL REPORT SUL DB");
            } else {
                // Costruzione del report
                report_string.append(">>> REPORT <<<\n"); (20)
                report_string.append("Periodo: " + dataInizio + " -> "
+ dataFine + "\n\n"); (21)
                report_string.append("Poesie pubblicate: " +
    report.getnPoesiePubblicate() +"\n"); (22)
                report_string.append("Tag più utilizzato: #"
+ report.getLeadTag() + "\n"); (23)
                report_string.append("Autore più prolifico: "
+ report.getLeadAutori() + "\n\n"); (24)
                report_string.append("Top 3 poesie con più like:\n"); (25)
                report_string.append(report.getLeadPoesie()); (26)
}

```



```

        }
        return report_string.toString(); (27)
    }
}

```

Il metodo analizzato richiama a sua volta i metodi `caricaListaDaDB()`, `convertiPoesieDAO()`, `calcoloTageAutore()`, `calcolaTopPoesie()` del livello Entity, dei quali dovremo valutare la complessità ciclomatica.

Numero Ciclomatico:

- Numero di regioni chiuse del grafo + 1 = 3
- Numero di nodi predicati (7, 17) + 1 = 3
- #archi - #nodi + 2 = (14 - 13) + 2 = 3

Cammini:

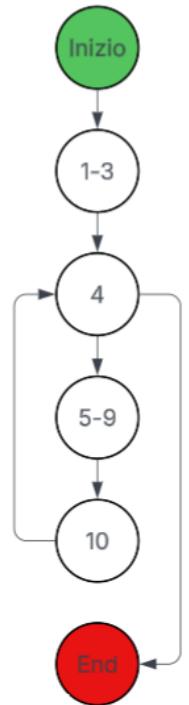
- 1) 1→(2-6)→7→(12-16)→17→18→19→27
- 2) 1→(2-6)→7→(8-10)→11→7→(12-16)→17→20→(21-25)→26→27
- 3) 1→(2-6)→7→(8-10)→11→7→(12-16)→17→18→19→27

6.1.1.2.1 caricaListaDaDB() - EntityPoetUp

```

public static void caricaListaDaDB() { //metodo per caricare la lista
degli studenti dal DB
    elencoUtenti = new ArrayList<>(); (1)
    UtenteDAO utente = new UtenteDAO(); (2)
    ArrayList<UtenteDAO> lista_db_utenti = utente.getListUtenti(); (+2
decision points)(3)
    for(int i = 0; i<lista_db_utenti.size();i++) { (4)
        EntityUtente utente_temp = new EntityUtente(); (5)
        utente_temp.setEmail(lista_db_utenti.get(i).getEmail()); (6)
        utente_temp.setPwd(lista_db_utenti.get(i).getPwd()); (7)
        utente_temp.setAmministratore
        (lista_db_utenti.get(i).isAmministratore()); (8)
        utente_temp.setId(lista_db_utenti.get(i).getId()); (9)
        elencoUtenti.add(utente_temp); (10)
    }
}

```

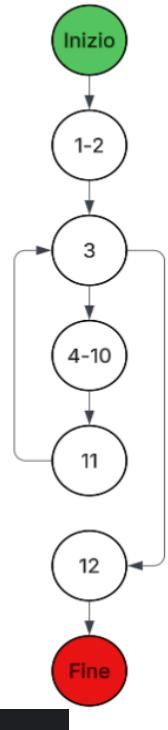


Numero Ciclomatico:

- Numero di regioni chiuse del grafo + 1 = 2
- Numero di nodi predicati (4) + 1 = 2
- #archi - #nodi + 2 = (4 - 4) + 2 = 2

Cammini:

- 1) (1-3)→4→(5-9)→10→4
- 2) (1-3)→4



6.1.1.2.2 convertiPoesieDAO - EntityAmministratore

```

private ArrayList<EntityPoesia> convertiPoesieDAO(ArrayList<PoesiaDAO>
tutteLePoesie) {
    ArrayList<EntityPoesia> poesie=new ArrayList<EntityPoesia>(); (1)
    EntityUtente autore = new EntityUtente(); (2)
    for (PoesiaDAO poesiaD : tutteLePoesie) { (3)
        EntityPoesia poesia_temp=new EntityPoesia(); (4)

        poesia_temp.setDatapubblicazione(poesiaD.getDatapubblicazione()); (5)
            autore.setId(poesiaD.getAutore()); (6)
            poesia_temp.setAutore(autore.getNickdaDB()); (7) (+2 decision
points)      poesia_temp.setTag(poesiaD.getTag()); (8)
            poesia_temp.setContatoreLike(poesiaD.getContatoreLike()); (9)
            poesia_temp.setTitolo(poesiaD.getTitolo()); (10)
            poesie.add(poesia_temp); (11)
    }
    return poesie; (12)
}

```

Numero Ciclomatico:

- Numero di regioni chiuse del grafo + 1 = 2
- Numero di nodi predicati (3) + 1 = 2
- #archi - #nodi + 2 = (5 - 5) + 2 = 2

Cammini:

- 1) (1-2)→3→(4-10)→11→3→12
- 2) (1-2)→3→12

6.1.1.2.3 calcoloTageAutore() - EntityAmministratore

```

public ArrayList<String> calcoloTageAutore(ArrayList<EntityPoesia> poesieDB, Date
dataInizio, Date dataFine) {
    // Dati per il report
    Map<String, Integer> tagCount = new HashMap<>(); (1)
    Map<String, Integer> autoreCount = new HashMap<>(); (2)
    ArrayList<EntityPoesia> poesieFiltrate = new ArrayList<>(); (3)
    for (EntityPoesia poesia : poesieDB) { (4)
        Date dataPoesia = poesia.getDatapubblicazione(); (5)
        // Filtro per intervallo di date
        if (dataPoesia != null && dataPoesia.compareTo(dataInizio) >= 0 &&
dataPoesia.compareTo(dataFine) <= 0) { (6)
            poesieFiltrate.add(poesia);
            // Conteggio tag
            String[] tags = poesia.getTag().split("#"); (7)
            for (String tag : tags) { (8)

```

```

        if (!tag.isBlank()) { (9)
            tag = tag.trim().toLowerCase(); (10)
            tagCount.put(tag, tagCount.getOrDefault(tag, 0) + 1); (11)
        }
    }
    // Conteggio poesie per autore
    String autore = poesia.getAutore(); (12)
    autoreCount.put(autore, autoreCount.getOrDefault(autore, 0) + 1); (13)
}
// Trova tag più utilizzato
String tagPiuUsato = tagCount.entrySet().stream()
    .max(Map.Entry.comparingByValue())
    .map(Map.Entry::getKey)
    .orElse("Nessun tag"); (14)
System.out.println(tagPiuUsato); (15)
// Trova autore più prolifico
String autoreTop = autoreCount.entrySet().stream()
    .max(Map.Entry.comparingByValue())
    .map(Map.Entry::getKey)
    .orElse("Nessun autore"); (16)
System.out.println(autoreTop); (17)
ArrayList<String> tagEautore=new ArrayList<String>(); (18)
tagEautore.add(autoreTop); (19)
tagEautore.add(tagPiuUsato); (20)
return tagEautore; (21)
}

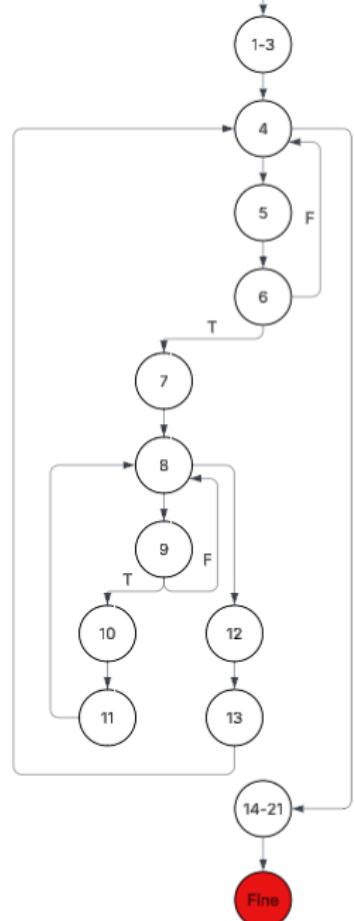
```

Numero Ciclomatico:

- Numero di regioni chiuse del grafo + 1 = 5
- Numero di nodi predicati (4, 6, 8, 9) + 1 = 5
- #archi - #nodi + 2 = (15 - 12) + 2 = 5

Cammini:

- 1) (1-3)→4→5→6→7→8→9→10→11→8→12→13→4→(14-21)
- 2) (1-3)→4→(14-21)
- 3) (1-3)→4→5→6→4→(14-21)
- 4) (1-3)→4→5→6→7→8→9→8→12→13→4→(14-21)
- 5) (1-3)→4→5→6→7→8→12→13→4→(14-21)

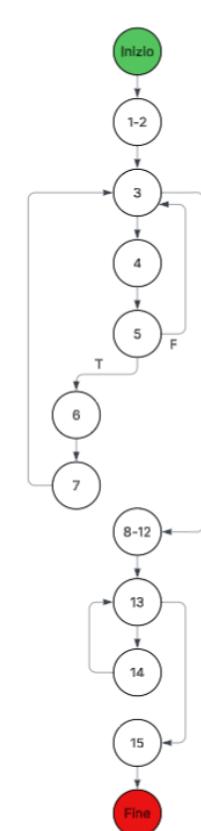


6.1.1.2.4 calcoloTopPoesie() - EntityAmministratore

```

private void calcoloTopPoesie(ArrayList<EntityPoesia> poesieDB, Date dataInizio,
Date dataFine, EntityReport report) {
    ArrayList<EntityPoesia> poesieFiltrate = new ArrayList<>(); (1)
    int contPoesie = 0; (2)
    for (EntityPoesia poesia : poesieDB) { (3)
        Date dataPoesia = poesia.getDataPubblicazione(); (4)
        // Filtro per intervallo di date
        if (dataPoesia != null && dataPoesia.compareTo(dataInizio) >= 0 &&
dataPoesia.compareTo(dataFine) <= 0) { (5)
            contPoesie++; (6)
            poesieFiltrate.add(poesia); (7)
        }
    }
    report.setnPoesiePubblicate(contPoesie); (8)
    poesieFiltrate.sort((p1, p2) -> Integer.compare(p2.getContatoreLike(),
p1.getContatoreLike())); (9)
    ArrayList<EntityPoesia> topPoesie = (ArrayList<EntityPoesia>)
poesieFiltrate.stream().limit(3).collect(Collectors.toList()); (10)
    StringBuilder leadBuilder = new StringBuilder(); (11)
    int posizione = 1; (12)
    for (EntityPoesia poesia : topPoesie) { (13)
        leadBuilder.append(posizione++ + ". \" " + poesia.getTitolo() + " "
di " + poesia.getAutore() + " (" + poesia.getContatoreLike() + " like)\n"); (14)
    }
    report.setLeadPoesie(leadBuilder.toString()); (15)
}

```



Numero Ciclomatico:

- Numero di regioni chiuse del grafo + 1 = 4
- Numero di nodi predicati (3, 5, 13) + 1 = 4
- #archi - #nodi + 2 = (12 - 10) + 2 = 4

Cammini:

- 1) (1-2) → 3 → 4 → 5 → 6 → 7 → 3 → (8-12) → 13 → 14 → 13 → 15
- 2) (1-2) → 3 → (8-12) → 13 → 15
- 3) (1-2) → 3 → 4 → 5 → 6 → 7 → 3 → (8-12) → 13 → 15
- 4) (1-2) → 3 → 4 → 5 → 3 → (8-12) → 13 → 15

Numero Ciclomatico complessivo: $3 + 2 + 2 + 5 + 4 - 1 * 4 + 7 = 19$

Il valore della complessità ciclomatica è stato incrementato di +7 poiché, analizzando i metodi interni richiamati dalle funzioni, sono emersi ulteriori

decision points. Accanto a ciascun metodo contenente tali punti decisionali è stato indicato il numero di unità da aggiungere al calcolo complessivo.

6.2 Test funzionale

6.2.1 Registrazione

TEST SUITE							
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese	Esito (FAIL, PASS)
1	Tutti input validi	Nickname valido, Email valida, Password valido		{Nickname="MarioRed02", Email: "MarioRossi02@gmail.com", Password: "Password1!"}	Utente registrato con successo	Il sistema mostra un messaggio per la registrazione avvenuta con successo e invia una notifica all'utente	PASS
2	Nickname stringa > 28 caratteri	Nickname stringa > 28 caratteri [ERROR], Email, Password validi		{Nickname="MarioRedddd dddddddddd d02", Email: "MarioRossi02@gmail.com", Password: "Password1!"}	Nome troppo lungo!		PASS
3	Nickname Stringa che contiene caratteri speciali	Nickname stringa con simboli [ERROR], Email, Password validi		{Nickname="M@rio-Red02", Email: "MarioRossi02@gmail.com", Password: "Password1!"}	Formato Nome errato!		PASS
4	Nickname stringa vuota	Nickname stringa vuota [ERROR], Email, Password validi		{Nickname="", Email: "MarioRossi02@gmail.com", Password: "Password1!"}	Errore Nickname vuoto!		PASS
5	Email Stringa di caratteri già registrati	Nickname valido, Email Stringa di caratteri già registrati [ERROR], Password valida		{Nickname="MarioRed02", Email: "MarioGiàReg@gmail.com", Password: "Password1!"}	Utente già registrato!		PASS
6	Email stringa di caratteri alfanumerici senza il simbolo @	Nickname valido, Email stringa di caratteri alfanumerici senza il simbolo @ [ERROR], Password valida		{Nickname="MarioRed02", Email: "MarioRossi02.gmail.com", Password: "Password1!"}	Formato Email errato!		PASS

7	Email stringa vuota	Nickname valido, Email stringa vuota [ERROR], Password valida		{Nickname="MarioRed02", Email: "", Password: "Password!"}	Tutti i campi sono obbligatori!		PASS
8	Password stringa senza caratteri speciali	Nickname, Email validi, Password stringa senza caratteri speciali [ERROR]		{Nickname="MarioRed02", Email: "MarioRossi02@gmail.com", Password: "Password!"}	Formato Password errato!		PASS
9	Password stringa di caratteri senza un numero	Nickname, Email validi, Password stringa di caratteri senza un numero [ERROR]		{Nickname="MarioRed02", Email: "MarioRossi02@gmail.com", Password: "Password!"}	Formato Password errato!		PASS
10	Password Stringa di caratteri alfanumerici di lunghezza <8	Nickname, Email validi, Password stringa di caratteri alfanumerici di lunghezza <8 [ERROR]		{Nickname="MarioRed02", Email: "MarioRossi02@gmail.com", Password: "Passw"}	Formato Password errato!		PASS
11	Password Stringa di caratteri alfanumerici di lunghezza >32	Nickname, Email validi, Password stringa di caratteri alfanumerici di lunghezza > 32 [ERROR]		{Nickname="MarioRed02", Email: "MarioRossi02@gmail.com", Password: "Passwordddddddddddddd ddddddcccccccccddddd ddd!!"}	Formato Password errato!		PASS
12	Password Stringa vuota	Nickname, Email validi, Password stringa vuota [ERROR]		{Nickname="MarioRed02", Email: "MarioRossi02@gmail.com", Password: ""}	Tutti i campi sono obbligatori!		PASS

6.2.2 Autenticazione

TEST SUITE							
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese	Esito (FAIL, PASS)
I	Tutti input validi	Email valida Password valida	Utente Registrato	{Email: “MarioRossi02@gmail.com”, Password: “Password1!”}	Accesso avvenuto con successo	Il sistema mostra un messaggio per l'autenticazione avvenuta	PASS

						con successo	
2	Email stringa di caratteri alfanumerici senza il simbolo @ [ERROR], Password valida	Email stringa di caratteri alfanumerici senza il simbolo @ [ERROR], Password valida		{Email: "MarioRossi02@gmail.com", Password: "Password1!"}	Formato Email errato!		PASS
3	Email Stringa di caratteri non registrata [ERROR], Password valida	Email stringa di caratteri già registrata [ERROR], Password valida		{Email: "Mario02NonReg@gmail.com", Password: "Password1!"}	Errore email non registrata!		PASS
4	Email Stringa vuota [ERROR], Password valida	Email Stringa vuota [ERROR], Password valida		{Email: "", Password: "Password1!"}	Inserisci email e password!		PASS
5	Password Stringa di caratteri non corrispondente alla password salvata [ERROR]	Email valida, Password stringa di caratteri non corrispondente a quella salvata [ERROR]		{Email: "MarioRossi02@gmail.com", Password: "PasswSbagl2?"}	La password inserita non è corretta!		PASS
6	Password stringa senza caratteri speciali [ERROR]	Email valida, Password stringa senza caratteri speciali [ERROR]		{Email: "MarioRossi02@gmail.com", Password: "Password1!"}	Formato Password errato!		PASS
7	Password stringa di caratteri senza un numero [ERROR]	Email valida, Password stringa di caratteri senza un numero [ERROR]		{Email: "MarioRossi02@gmail.com", Password: "Password@!"}	Formato Password errato!		PASS
8	Password Stringa di caratteri alfanumerici di lunghezza <8 [ERROR]	Email valida, Password stringa di caratteri alfanumerici di lunghezza <8 [ERROR]		{Email: "MarioRossi02@gmail.com", Password: "Passw"}	Formato Password errato!		PASS
9	Password Stringa di caratteri alfanumerici di lunghezza >32 [ERROR]	Email valida, Password stringa di caratteri alfanumerici di lunghezza > 32 [ERROR]		{Email: "MarioRossi02@gmail.com", Password: "Passwordddddddddddddd ddddddcccccccccddddd ddd!!"}	Formato Password errato!		PASS

10	Password Stringa vuota	Email valida, Password Stringa vuota [ERROR]		{Email: “MarioRossi02@gmail.com”, Password: “”}	Inserisci email e password		PASS
----	---------------------------	---	--	--	----------------------------------	--	-------------

6.2.3 Gestione Profilo Personale

TEST SUITE							
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese	Esito (FAIL, PASS)
1	Tutti input validi	Nome valido, Cognome valido, DataNascita valida Biografia valida	Utente registrato	{Nome=“Mario”, Cognome=“Rossi”, DataNascita=“2003-21-11”, Biografia=“Giovane aspirante poeta”}	Profilo aggiornato con successo!	Il sistema mostra il messaggio di modifica avvenuta con successo	PASS
2	Nome Stringa di caratteri di lunghezza > 40 [ERROR], Cognome, DataNascita, Biografia validi	Nome Stringa di caratteri di lunghezza > 40 [ERROR], Cognome, DataNascita, Biografia validi		{Nome=“Marioooooooooooooooo”, Cognome=“Rossi”, DataNascita=“2003-21-11”, Biografia=“Giovane aspirante poeta”}	Nome troppo lungo!		PASS
3	Nome Stringa che contiene caratteri speciali	Nome Stringa che contiene simboli che non sono caratteri [ERROR], Cognome, DataNascita Biografia validi		{Nome=“M@rio”, Cognome=“Rossi”, DataNascita=“2003/21/11”, Biografia=“Giovane aspirante poeta”}	Formato nome errato!		PASS
4	Cognome Stringa di caratteri di lunghezza > 40 [ERROR], DataNascita, Biografia validi	Nome valido, Cognome Stringa di caratteri di lunghezza > 40 [ERROR], DataNascita, Biografia validi		{Nome=“Mario”, Cognome=“Rossi ”, DataNascita=“2003-21-11”, Biografia=“Giovane aspirante poeta”}	Cognome troppo lungo!		PASS
5	Cognome Stringa che contiene caratteri speciali	Nome valido, Cognome Stringa che contiene caratteri speciali [ERROR], DataNascita, Biografia validi		{Nome=“Mario”, Cognome=“#Ro-ssi”, DataNascita=“2003-21-11”, Biografia=“Giovane aspirante poeta”}	Formato Cognome errato!		PASS

6	DataNascita successiva a quella odierna	Nome valido, Cognome valido, DataNascita successiva a quella odierna [ERROR] Biografia valida		{Nome="Mario", Cognome="Rossi", DataNascita="2100-21-11", Biografia="Giovane aspirante poeta"}	Errore data nel futuro!		PASS
7	Formato DataNascita non valido	Nome valido, Cognome valido, Formato DataNascita non valido [ERROR], Biografia valida		{Nome="Mario", Cognome="Rossi", DataNascita="21-11-2003", Biografia="Giovane aspirante poeta"}	Formato DataNascita errato!		PASS
8	Biografia Stringa di caratteri di lunghezza > 250	Nome, Cognome, DataNascita validi, Biografia Stringa di caratteri di lunghezza > 255 [ERROR]		{Nome="Mario", Cognome="Rossi", DataNascita="2003-21-11", Biografia="Giovane aspirante poetaaaaaa(x256)"} Biografia troppo lunga!			PASS

Nella prima stesura dei test si è pensato di inserire l'immagine tramite Url, in fase di codifica è stato deciso di inserirla tramite scelta da immagini già presenti nel database.

6.2.4 Pubblicazione Poesia

TEST SUITE									
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese	Output Ottenuati	Post-condizioni Ottenute	Esito (FAIL, PASS)
1	Tutti input validi e raccolta esistente	Titolo valido, Testo valido, Tag valido, Raccolta valida	Raccolta di destinazione esistente	{Titolo="Mattina", Testo="M'illumino d'immenso", Tag="#Gioia", Raccolta="Allegria"}	Poesia creata e aggiunta alla raccolta	Il sistema mostra il messaggio di creazione avvenuta con successo e aggiunge la poesia alla raccolta	Poesia Pubblicata!	Poesia creata e aggiunta alla raccolta	PASS
2	Tutti input validi e raccolta inesistente	Titolo valido, Testo valido, Tag valido, Raccolta valida ma non		{Titolo="Soldati", Testo="Si sta come d'autunno sugli alberi le foglie", Tag="#Tristezza",}	Raccolta creata e poesia aggiunta ad essa	Il sistema mostra il messaggio di creazione avvenuta con	Poesia Pubblicata!	Raccolta creata e poesia aggiunta ad essa	PASS

		esistente [EXTEND: CreazioneRa ccolta]		Raccolta="Preferiti "} successo, crea la raccolta e vi aggiunge la poesia				
3	Titolo Stringa di caratteri di lunghezz a > 25	Titolo Stringa di caratteri di lunghezza > 25 [ERROR], Testo, Tag, Raccolta validi		{Titolo="Mattinaaa aaaaaaaaaaaaaaaaaaa" , Testo="M'illumino d'immenso", Tag="#Gioia", Raccolta="Allegria "}	Titolo troppo lungo!	-		PASS
4	Titolo Stringa che contiene caratteri speciali	Titolo Stringa che contiene caratteri speciali [ERROR], Testo, Tag, Raccolta validi		{Titolo="M@ttina ", Testo="M'illumino d'immenso", Tag="#Gioia", Raccolta="Allegria "}	Formato Titolo errato!	-		PASS
5	Titolo Stringa vuota	Titolo Stringa vuota [ERROR], Testo, Tag, Raccolta validi		{Titolo="", Testo="M'illumino d'immenso", Tag="#Gioia", Raccolta="Allegria "}	Il Titolo non può essere vuoto!	-		PASS
6	Testo Stringa di caratteri di lunghezz a > 500	Titolo valido, Testo Stringa di caratteri di lunghezza > 500 [ERROR], Tag, Raccolta validi		{Titolo="Mattina", Testo="M'illumino d'immensoooo(x5 0!)", Tag="#Gioia", Raccolta="Allegria "}	Testo troppo lungo!	-		PASS
7	Testo Stringa che contiene caratteri speciali che non sono segni di punteggiatur a	Titolo valido, Testo Stringa che contiene caratteri speciali che non sono segni di punteggiatur a [ERROR], Tag, Raccolta validi		{Titolo="Mattina", Testo="#M'illumin o d'immenso", Tag="#Gioia", Raccolta="Allegria "}	Formato Testo errato!	-		PASS

8	Testo Stringa vuota	Titolo valido, Testo Stringa vuota [ERROR], Tag, Raccolta validi		{Titolo="Mattina", Testo="", Tag="#Gioia", Raccolta="Allegria"} "	Il Testo non può essere vuoto	-			PASS
9	Tag Stringa di caratteri di lunghezz a > 255	Titolo valido, Testo Valido, Tag Stringa di caratteri di lunghezza > 255 [ERROR], Raccolta valida		{Titolo="Mattina", Testo="M'illumino d'immenso", Tag="#Gioiaa...("x256)", Raccolta="Allegria"} "	Tag troppo lungo!	-			PASS
10	Tag Stringa che contiene caratteri speciali al di fuori di #	Titolo valido, Testo valido, Tag Stringa che contiene caratteri speciali al di fuori di # [ERROR], Raccolta valida		{Titolo="Mattina", Testo="M'illumino d'immenso", Tag="#Gioi@", Raccolta="Allegria"} "	Formato Tag errato!	-			PASS
11	Tag Stringa vuota	Titolo valido, Testo valido, Tag Stringa vuota [ERROR], Raccolta valida		{Titolo="Mattina", Testo="M'illumino d'immenso", Tag="", Raccolta="Allegria"} "	Il Tag non può essere vuoto!	-			PASS
12	Raccolta Stringa di caratteri di lunghezz a > 25	Titolo, Testo, Tag validi, Raccolta Stringa di caratteri di lunghezza > 25 [ERROR]		{Titolo="Mattina", Testo="M'illumino d'immenso", Tag="#Gioia", Raccolta="Allegriaaaaaaaaaaaaaaaaaaaaaaaa"} "	Raccolta troppo lunga!	-			PASS
13	Raccolta Stringa che contiene caratteri speciali	Titolo, Testo, Tag validi, Raccolta Stringa che contiene caratteri speciali [ERROR]		{Titolo="Mattina", Testo="M'illumino d'immenso", Tag="#Gioia", Raccolta="Allegri@"} "	Formato Raccolta non valido!	-			PASS

14	Raccolta Stringa vuota	Titolo, Testo, Tag validi, Raccolta Stringa vuota	[ERROR]	{Titolo="Mattina", Testo="M'illumino d'immenso", Tag="#Gioia", Raccolta=""}	Raccolta non può essere vuoto	-			PASS

6.2.5 CreazioneRaccolta

TEST SUITE							
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese	Esito (FAIL, PASS)
1	Tutti input validi	Titolo valido, Descrizione valida		{Titolo="Allegria", Descrizione="L'Allegria è una raccolta di poesie di Giuseppe Ungaretti"}	La Raccolta viene creata		PASS
2	Titolo Stringa di caratteri > 25 [ERROR], Descrizione valida	Titolo Stringa di caratteri > 25 [ERROR], Descrizione valida		{Titolo="Allegriaaaaaaaaaaaaaaaaaaaa", Descrizione="L'Allegria è una raccolta di poesie di Giuseppe Ungaretti"}	Titolo troppo lungo!		PASS
3	Titolo Stringa che contiene simboli che non sono caratteri [ERROR], Descrizione valida	Titolo Stringa che contiene simboli che non sono caratteri [ERROR], Descrizione valida		{Titolo="Allegri@", Descrizione="L'Allegria è una raccolta di poesie di Giuseppe Ungaretti"}	Formato Titolo errato!		PASS
4	Titolo Stringa vuota	Titolo Stringa vuota [ERROR], Descrizione valida		{Titolo="", Descrizione="L'Allegria è una raccolta di poesie di Giuseppe Ungaretti"}	Il Titolo non può essere vuoto		PASS
5	Descrizione Stringa di caratteri > 255 [ERROR]	Titolo valido, Descrizione Stringa di caratteri > 255 [ERROR]		{Titolo="Allegria", Descrizione="L'Allegria è una raccolta di poesie di Giuseppe Ungarettioooooooooooo... (x256)"}	Descrizione troppo lunga!		PASS

6.2.6 SpostaPoesiaInRaccolta

TEST SUITE							
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese	Esito (FAIL, PASS)
I	Tutti input validi	RaccoltaDestinazione valida		{RaccoltaDestinazione="Allegria"}	Poesia spostata	La poesia viene rimossa	PASS

					con Successo!	dalla raccolta di partenza e aggiunta alla raccolta di destinazione	
2	RaccoltaDestinazione Inesistente	RaccoltaDestinazione valido ma Raccolta Inesistente [ERROR]		{RaccoltaDestinazione="Racc NonEsistente"}	Raccolta inesistente!		PASS
3	RaccoltaDestinazione Stringa di caratteri di lunghezza > 25	RaccoltaDestinazione Stringa di caratteri di lunghezza > 25 [ERROR]		{RaccoltaDestinazione="Allegraaaaaaaaaaaaaaaaaaaaaaa"} aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa	Titolo troppo lungo!		PASS
4	RaccoltaDestinazione Stringa che contiene caratteri speciali	RaccoltaDestinazione Stringa che contiene caratteri speciali [ERROR]		{RaccoltaDestinazione="Allegr i@"}	Formato Raccolta destinazione errato!		PASS
5	RaccoltaDestinazione Stringa vuota	RaccoltaDestinazione Stringa vuota [ERROR]		{RaccoltaDestinazione=""}	Raccolta di destinazione non inserita!		PASS

6.2.7 ModificaRaccolta

TEST SUITE							
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese	Esito (FAIL, PASS)
1	Tutti input validi	Titolo valido, Descrizione valida		{Titolo="Allegria", Descrizione="L'Allegria" è una raccolta di poesie di Giuseppe Ungaretti"}	La Raccolta viene modificata		PASS
2	Titolo Stringa di caratteri > 25 [ERROR], Descrizione valida	Titolo Stringa di caratteri > 25 [ERROR], Descrizione valida		{Titolo="Allegriaaaaaaaaaaaaaaaaaaaaaaaa", Descrizione="L'Allegria" è una raccolta di poesie di Giuseppe Ungaretti"}	Titolo troppo lungo!		PASS
3	Titolo Stringa che contiene caratteri speciali	Titolo Stringa che contiene caratteri speciali [ERROR], Descrizione valida		{Titolo="Allegri@", Descrizione="L'Allegria" è una raccolta di poesie di Giuseppe Ungaretti"}	Formato Titolo errato!		PASS

4	Titolo Stringa vuota	Titolo Stringa vuota [ERROR], Descrizione valida		{Titolo="”, Descrizione=”L’Allegria” è una raccolta di poesie di Giuseppe Ungaretti”}	Il Titolo non può essere vuoto		PASS
5	Descrizione Stringa di caratteri > 255	Titolo valido, Descrizione Stringa di caratteri > 255 [ERROR]		{Titolo=”Allegria”, Descrizione=”L’Allegria” è una raccolta di poesie di Giuseppe Ungaretti……(x256) ”}	Descrizione troppo lunga!		PASS

6.2.8 commentaPoesia

TEST SUITE							
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese	Esito (FAIL, PASS)
1	Tutti input validi	Testo valido		{Testo=”Mi sono emozionato leggendo questa poesia, semplicemente sensazionale, grazie per averla condivisa”}	Commento postato con successo!	Il commento viene inserito nella sezione commenti della poesia di riferimento	PASS
2	Testo Stringa di caratteri di lunghezza > 255	Testo Stringa di caratteri di lunghezza > 255 [ERROR]		{Testo=”Mi sono emozionato leggendo questa poesia, semplicemente sensazionale, grazie per averla condivisa…(x256)”}	Testo troppo lungo!		PASS
3	Testo Stringa vuota	Testo Stringa vuota [ERROR]		{Testo=””}	Testo vuoto!		PASS

6.2.9 GeneraReport

TEST SUITE							
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese	Esito (FAIL, PASS)
1	Tutti input validi	DataInizio valida, DataFine valida		{DataInizio=”2010-05-10”, DataFine=”2010-05-17”}	Report generato con successo!	Il sistema genera i dati del report e li mostra sull’interfaccia	PASS

2	DataInizio successiva a quella odierna	DataInizio successiva a quella odierna [ERROR], DataFine valida		{DataInizio="2100-05-10", DataFine="2010-05-17"}	DataInizio errata!		PASS
3	DataInizio con formato non valido	DataInizio con formato non valido [ERROR], DataFine valida		{DataInizio="10-05-2010", DataFine="2010-05-17"}	Formato DataInizio non valido!		PASS
4	DataFine inferiore a DataInizio	DataInizio valida, DataFine inferiore a DataInizio [ERROR]		{DataInizio="2010-05-10", DataFine="2009-05-17"}	La DataFine non può essere inferiore alla DataInizio!		PASS
5	DataFine successiva a quella odierna	DataInizio valida, DataFine successiva a quella odierna [ERROR]		{DataInizio="2010-05-10", DataFine="2100-05-17"}	DataFine errata!		PASS
6	DataFine con formato non valido	DataInizio valida, DataFine con formato non valido [ERROR]		{DataInizio="2010-05-10", DataFine="17-05-2010"}	Formato DataFine non valido!		PASS

6.2.10 RicercaPoesiaPerTag

TEST SUITE							
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese	Esito (FAIL, PASS)
1	Tutti input validi	RicercaTag valida		{RicercaTag="Amore"}	Mostra Risultati ricerca!		PASS
2	RicercaTag Stringa di caratteri di lunghezza > 255 [ERROR]	RicercaTag Stringa di caratteri di lunghezza > 255 [ERROR]		{RicercaTag="Amoreeeeeeeeeeee...((x256))"}	Ricerca troppo lunga!		PASS
3	RicercaTag Stringa che contiene caratteri speciali al di fuori di # [ERROR]	RicercaTag Stringa che contiene caratteri speciali al di fuori di # [ERROR]		{RicercaTag="#Amore@!"}	Formato Ricerca errato!		PASS
4	RicercaTag Stringa vuota [ERROR]	RicercaTag Stringa vuota [ERROR]		{RicercaTag=""}	La Ricerca non può		PASS

					essere vuota!		
--	--	--	--	--	---------------	--	--

6.2.11 RicercaPoesiaPerTesto

TEST SUITE							
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese	Esito (FAIL, PASS)
1	Tutti input validi	RicercaTesto valida		{RicercaTesto="M'illumino D'immenso"}	Mostra Risultati ricerca!		PASS
2	RicercaTesto Stringa di caratteri di lunghezza > 255	RicercaTesto Stringa di caratteri di lunghezza > 500 [ERROR]		{RicercaTesto="M'illumino D'immenso... (x256)"}	Ricerca troppo lunga!		PASS
3	RicercaTesto Stringa che contiene caratteri che non sono segni di punteggiatura [ERROR]	RicercaTesto Stringa che contiene caratteri che non sono segni di punteggiatura [ERROR]		{RicercaTesto="M'illumino D'immenso@"}	Formato Ricerca errato!		PASS
4	RicercaTesto Stringa vuota [ERROR]	RicercaTesto Stringa vuota [ERROR]		{RicercaTesto=""}	La Ricerca non può essere vuota!		PASS

6.2.12 RicercaPoesiaPerTitolo

TEST SUITE							
Test Case ID	Descrizione	Classi di equivalenza coperte	Pre-condizioni	Input	Output attesi	Post-condizioni attese	Esito (FAIL, PASS)
1	Tutti input validi	RicercaTitolo valida		{RicercaTitolo="Mattina"}	Mostra Risultati ricerca!		PASS
2	RicercaTitolo Stringa di caratteri di lunghezza > 25	RicercaTitolo Stringa di caratteri di lunghezza > 25 [ERROR]		{RicercaTitolo="Mattinaaaaaaaaaaaaaaaaaaaaaaaaaaaaa"}	Ricerca troppo lunga!		PASS
3	RicercaTitolo Stringa che contiene caratteri speciali [ERROR]	RicercaTitolo Stringa che contiene caratteri speciali [ERROR]		{RicercaTitolo="M@ttina"}	Formato Ricerca errato!		PASS

4	RicercaTitolo Stringa vuota	RicercaTitolo Stringa vuota [ERROR]		{RicercaTitolo=""}	La Ricerca non può essere vuota!		PASS
---	-----------------------------	-------------------------------------	--	--------------------	----------------------------------	--	------